

Enumerating all maximal biclusters in numerical datasets

Rosana Veroneze^{a,*}, Arindam Banerjee^b, Fernando J. Von Zuben^a

^aUniversity of Campinas (DCA/FEEC), 400 Albert Einstein Street, Campinas, SP, Brazil

^bUniversity of Minnesota, 200 Union Street, Minneapolis, MN, USA

Abstract

Biclustering has proved to be a powerful data analysis technique due to its wide success in various application domains. However, the existing literature presents efficient solutions only for enumerating maximal biclusters with constant values, or heuristic-based approaches which can not find all biclusters or even support the maximality of the obtained biclusters. Here, we present a general family of biclustering algorithms for enumerating all maximal biclusters with (i) constant values on rows, (ii) constant values on columns, or (iii) coherent values. Versions for perfect and for perturbed biclusters are provided. Our algorithms have four key properties (just the algorithm for perturbed biclusters with coherent values fails to exhibit the first property): they are (1) efficient (take polynomial time per pattern), (2) complete (find all maximal biclusters), (3) correct (all biclusters attend the user-defined measure of similarity), and (4) non-redundant (all the obtained biclusters are maximal and the same bicluster is not enumerated twice). They are based on a generalization of an efficient formal concept analysis algorithm called In-Close2. Experimental results point to the necessity of having efficient enumerative biclustering algorithms and provide a valuable insight into the scalability of our family of algorithms and its sensitivity to user-defined parameters.

Keywords: Biclustering, formal concept analysis, frequent pattern mining, maximal bicliques, data mining.

1. Introduction

Biclustering is a local approach for clustering that operates simultaneously over the set of objects and attributes of a data matrix. It looks for submatrices constituted of subsets of objects that have a highly consistent pattern across a subset of attributes. Biclustering methods are thus able to consider coherence measures which are more general than distance functions, such as Euclidean and Manhattan distances, and hence are going to find biclusters supporting more general affinities than conventional numerical proximity of elements [73].

*Corresponding author

Email address: veroneze@dca.fee.unicamp.br (Rosana Veroneze)

In the literature, biclustering has great value in finding interesting patterns in microarray expression data [78]. Indeed, the application of biclustering is fully disseminated and not limited to biological data. For instance, we can mention: dimensionality reduction [2], text mining [10, 24], collaborative filtering [5, 21, 68, 69], and treatment of missing data [10, 20, 22, 71]. Moreover, the importance of biclustering continues to increase, as researchers are (i) finding new applications in scientific and commercial domains, including bioinformatics, social network analysis, and text mining; and (ii) unveiling the connection between biclustering and several other important problems, including subspace clustering [46], frequent pattern mining (FPM) [31], and formal concept analysis (FCA) [28].

Biclustering may be interpreted as a hard combinatorial optimization problem. The more flexible the bicluster structure, the more complex the problem, and we are considering the most flexible structure in this work: arbitrarily positioned overlapping biclusters [56]. Thereby, an object/attribute can belong to none, one, or more than one bicluster. In this scenario, finding all maximal biclusters in a data matrix is an NP-hard problem [57]. Due to this, most of the proposed algorithms are heuristic-based [56], and many of them consider an inflexible bicluster structure and mine a number of biclusters defined a priori. The heuristics-based algorithms potentially produce sub-optimal solutions, missing important biclusters and not guaranteeing the maximality of the identified ones. Some examples of well-known heuristic-based biclustering algorithms are: CC [16], FLOC [74], ROCC [23], ISA [37], Plaid [50], and OPSM [11]. For surveys, refer to Madeira and Oliveira [56] and Busygin et al. [14].

In FCA and related areas, such as FPM and graph theory, we have plenty of algorithms for enumerating all maximal biclusters with constant values (CTV) in a binary dataset. These maximal CTV biclusters are called formal concepts in FCA, closed frequent itemsets (or patterns) in FPM¹, and maximal bicliques in graph theory (for more details about the connection of these areas, see Section 3.2). Some examples of algorithms are: Makino and Uno [58], Eppstein *et al.* [25], Close-by-One (CbO) [48], In-Close [6], In-Close2 [7], FCbO [45], CHARM [76], and LCM [70]. Their enumeration process is characterized by being:

1. Efficient: it takes polynomial time per pattern, i.e., it takes polynomial time to enumerate the first bicluster and takes polynomial time between enumerating two consecutive biclusters. It is the best one can computationally do in such scenario. If done properly, such algorithm will have time complexity linear in the number of biclusters and polynomial in the input size. Moreover, if the number of maximal biclusters is polynomial in the input size, the overall algorithm will be a polynomial time algorithm.
2. Complete: it finds all maximal biclusters. A complete enumeration guarantees to include the results produced by any other biclustering solution (given the same restrictions of similarity and

¹Being more specific, a closed frequent itemset corresponds to the column-set of a bicluster.

size). So, such biclustering solution is at least of equal quality, but probably of better quality, when compared with the solution provided by any other contender.

3. Correct: all found biclusters attend the user-defined measure of similarity. For instance, in the case of the aforementioned algorithms, all biclusters are submatrices of ones. Complete and correct enumerators are crucial for some applications such as the identification of biological indicators [54] and classification based on associations [53].
4. Non-redundant: all biclusters are maximal and it does not enumerate the same maximal bicluster twice. Property number 4 is very important because the number of biclusters produced from a dataset can be very large. So, it is useful to identify the smallest representative set of biclusters from which all other biclusters can be derived [65]. The set of all maximal biclusters is necessary and sufficient to capture all the information about the biclusters, and has a much smaller cardinality than the set of all attainable biclusters [75]. It is important to note that the algorithm must have a smart solution to avoid redundancy, otherwise it will not be efficient. For instance, a procedure to be avoid is to check if a new bicluster is not redundant by comparing with all previously mined biclusters.

Once the researchers found the link between these areas and biclustering, many algorithms have been proposed to deal with numerical (not only binary, but also integer or real-valued) datasets and other types of biclusters. In fact, nowadays the state-of-the-art biclustering algorithms are based on FPM [35]. Many proposals, such as [35, 57, 59, 61, 62, 67], binarize the data and apply the aforementioned algorithms. However, binarizing the dataset leads to loss of information, and guides to the necessity of tedious Boolean property encoding phases [12]. Therefore, there are also proposals to deal directly with numerical datasets, such as [12, 41, 64, 66, 78]. Without binarizing the numerical dataset, we are going to show in Section 4 that there is no proposal in the literature able to enumerate biclusters with (i) constant value on columns (CVC), (ii) constant values on rows (CVR), or (iii) coherent values (CHV) (see definitions in Section 2), so that the aforementioned four properties are preserved in this extended scenario (not only binary values in the dataset). Although some authors claim that their proposals do preserve these four properties, a more careful analysis to be presented in Section 4 shows that they all fail to exhibit one or more of these four properties. So, the aim of this paper is to cover some of these gaps. In fact, we are proposing algorithms capable of preserving these four properties when enumerating perfect CVC biclusters, perturbed CVC biclusters, and perfect CHV biclusters. The problem of enumerating CVR biclusters is equivalent to enumerating CVC biclusters. We are also proposing an algorithm with the last three of these properties to enumerate perturbed CHV biclusters. Note that CVC, CVR and CHV biclusters are a generalization of CTV biclusters [56] (for more details see Section 2.2). We call our family of algorithms *RIn-Close* because they are

generalizations of the FCA algorithm In-Close2 [7]. Tables 1 and 2 (see Section 4) show a technical comparison between our proposals and the competitors, attesting that we are proposing a number of improvements when enumerating biclusters from numerical datasets.

The remainder of the paper is organized as follows. Section 2 introduces definitions and mathematical formulation for biclustering. Section 3 reviews FCA and the algorithm In-Close2. Section 4 is devoted to related works. Section 5 presents the main contributions of this paper, more specifically the RIn-Close family of efficient enumerative algorithms for maximal CVC, CVR, or CHV biclusters. Experimental results are discussed in Section 6, and we conclude in Section 7.

2. Biclustering

The term *biclustering* was introduced by Mirkin [60] to describe the simultaneous clustering of the sets of rows and columns of a data matrix. More recently, the term was used in the analysis of gene expression data [16]. Cheng and Church [16] were responsible for the popularization of biclustering techniques with their algorithm called CC. However, Hartigan [34] was the first one to propose an algorithm for biclustering, using the term *direct clustering*. Other terms that are found in literature are: co-clustering, two-way clustering and bidimensional clustering, among others [56].

2.1. Definitions and Taxonomy of Biclusters

Let $\mathbf{A}_{n \times m}$ be a data matrix with the row index set $X = \{1, 2, \dots, n\}$ and the column index set $Y = \{1, 2, \dots, m\}$. Each element $a_{ij} \in \mathbf{A}$ represents the relationship between row i and column j . We use (X, Y) to denote the entire matrix \mathbf{A} . Considering that $I \subseteq X$ and $J \subseteq Y$, $\mathbf{A}_{IJ} = (I, J)$ denotes the submatrix of \mathbf{A} with the row index subset I and column index subset J .

Definition 2.1. A bicluster is a submatrix (I, J) of the data matrix $\mathbf{A}_{n \times m}$ such that the rows in the index subset $I = \{i_1, \dots, i_k\}$ ($I \subseteq X$ and $k \leq n$) exhibits similar behavior across the columns in the index subset $J = \{j_1, \dots, j_s\}$ ($J \subseteq Y$ and $s \leq m$), and vice-versa.

Thus, a bicluster (I, J) is a $k \times s$ submatrix of the matrix \mathbf{A} , with not necessarily contiguous rows and columns, such that it meets a certain homogeneity criterion. A biclustering algorithm looks for a set of biclusters $\mathfrak{B} = (I_l, J_l), l = 1, \dots, q$, such that each bicluster (I_l, J_l) satisfies some specific characteristics of homogeneity [56]. Considering these characteristics, there are four major types of biclusters [56]: (i) biclusters with constant values (CTV), (ii) biclusters with constant values on columns (CVC) or rows (CVR), (iii) biclusters with coherent values (CHV), and (iv) biclusters with coherent evolutions (CHE). The total number of biclusters, q , will depend on the features of the selected biclustering algorithm, on the constraints imposed, and on the behavior of the dataset being analyzed.

In this paper, we will provide the definition of CTV, CVC, CVR, and CHV biclusters. Please, refer to Madeira and Oliveira [56] for a complete survey with definitions and examples of all bicluster types.

2.2. Types of Biclusters

Although perfect biclusters can be found in some data matrices, in real data, they are usually masked by noise. Therefore, we will define the perfect and the perturbed case for all types of biclusters. The perturbed case is always a generalization of the perfect case. An user-defined parameter $\epsilon \geq 0$ determines the maximum residue (perturbation) allowed in a bicluster.

Definition 2.2 (CTV biclusters). *A perfect CTV bicluster is a submatrix (I, J) of a data matrix $\mathbf{A}_{n \times m}$ such that $a_{ij} = a_{kl}$, $\forall i, k \in I$ and $\forall j, l \in J$. A perturbed CTV bicluster is a submatrix (I, J) of a data matrix $\mathbf{A}_{n \times m}$ such that $|a_{ij} - a_{kl}| \leq \epsilon$, $\forall i, k \in I$ and $\forall j, l \in J$, i.e.,*

$$\max_{i \in I, j \in J} (a_{ij}) - \min_{i \in I, j \in J} (a_{ij}) \leq \epsilon. \quad (1)$$

Definition 2.3 (CVC biclusters). *A perfect CVC bicluster is a submatrix (I, J) of a data matrix $\mathbf{A}_{n \times m}$ such that $a_{ij} = a_{kj}$, $\forall i, k \in I$ and $\forall j \in J$. A perturbed CVC bicluster is a submatrix (I, J) such that $|a_{ij} - a_{kj}| \leq \epsilon$, $\forall i, k \in I$ and $\forall j \in J$, i.e.,*

$$\max_{i \in I} (a_{ij}) - \min_{i \in I} (a_{ij}) \leq \epsilon, \forall j \in J. \quad (2)$$

The definition of a *CVR bicluster* is the equivalent transpose of the definition of a CVC bicluster.

There are two perspectives for CHV biclusters: (i) additive model, and (ii) multiplicative model. Biclusters based on the additive model are called *shifting biclusters*. Biclusters based on the multiplicative model are called *scaling biclusters*. Any row (column) of a perfect shifting bicluster can be obtained by adding a constant value to any other row (column) of the bicluster. Similarly, any row (column) of a perfect scaling bicluster can be obtained by multiplying a constant value to any other row (column) of the bicluster. The problems of mining shifting and scaling biclusters are equivalent. Using an algorithm to mine shifting (scaling) biclusters, we can mine scaling (shifting) biclusters by previously taking the logarithm (exponential) of all entries of the data matrix (see a proof in Zhao and Zaki [78]). Therefore, we are going to focus only in the additive model in this paper. Another interesting property of mining CHV biclusters is that it is a symmetric problem. The CHV biclusters are fully preserved when rows become columns and columns become rows of the matrix (see a proof in Zhao and Zaki [78]).

Definition 2.4 (CHV biclusters - additive model). *Let $Z^{jl} = \{a_{ij} - a_{il}\}_{i \in I}$, $j, l \in J$, i.e., the set of values of the difference between two attributes for the subset of rows I . A perfect shifting bicluster is a submatrix (I, J) of a data matrix $\mathbf{A}_{n \times m}$ such that all elements of the set Z^{jl} , $\forall j, l \in J$, are equal, i.e., $z = w$, $\forall z, w \in Z^{jl}$, $\forall j, l \in J$. A perturbed shifting bicluster is a submatrix (I, J) such that $|z - w| \leq \epsilon$, $\forall z, w \in Z^{jl}$, $\forall j, l \in J$, i.e.,*

$$\max(Z^{jl}) - \min(Z^{jl}) \leq \epsilon, \forall j, l \in J. \quad (3)$$

2.3. Metrics and indices

Here, we will outline some bicluster metrics and indices to ease the reading of this work.

The *volume* of a bicluster (I, J) is given by $|I| \times |J|$. The *overlap* between two biclusters (I, J) and (I', J') is given by

$$ove((I, J), (I', J')) = \frac{|I \cap I'| \times |J \cap J'|}{\min(|I \times J|, |I' \times J'|)}. \quad (4)$$

Let $\mathfrak{B} = (I_l, J_l)$, $l = 1, \dots, k$, be a biclustering solution. The *span* of the solution \mathfrak{B} is given by

$$span(\mathfrak{B}) = \bigcup_{(I_l, J_l)} I_l \times J_l. \quad (5)$$

The *coverage* of the solution \mathfrak{B} is given by $cov(\mathfrak{B}) = |span(\mathfrak{B})|$, i.e., the number of cells of the data matrix covered by at least one bicluster. It is more usual to present the coverage in terms of percentage, i.e., $cov(\mathfrak{B})/(n \times m)$. The *global overlap* of the solution \mathfrak{B} is given by

$$oveg(\mathfrak{B}) = \frac{\sum_{(I_l \times J_l)} |I_l \times J_l| - cov(\mathfrak{B})}{cov(\mathfrak{B})}. \quad (6)$$

If we have a reference bicluster solution \mathfrak{B} , we can measure how good is a found bicluster solution \mathfrak{B} by means of an external evaluation [36]. We will use *precision* and *recall* to this end, respectively given by $prec(\mathfrak{B}, \mathfrak{B}) = |span(\mathfrak{B}) \cap span(\mathfrak{B})|/cov(\mathfrak{B})$ and $rec(\mathfrak{B}, \mathfrak{B}) = prec(\mathfrak{B}, \mathfrak{B})$.

2.4. Maximality and Monotonicity

Definition 2.5 (Maximal bicluster). *Given the desired characteristics of homogeneity, a bicluster (I, J) is called a maximal bicluster iff:*

- $\forall x \in X \setminus I$, $(I \cup \{x\}, J)$ is not a (valid) bicluster, and
- $\forall y \in Y \setminus J$, $(I, J \cup \{y\})$ is not a (valid) bicluster.

It means that a bicluster is maximal if we can not add any object/attribute to it without violating the desired characteristics of homogeneity.

For all bicluster definitions given in Subsection 2.2, we have the following properties.

Property 2.1 (Anti-Monotonicity). *Let (I, J) be a bicluster. Any submatrix (I', J') , where $I' \subseteq I$ and $J' \subseteq J$, is also a bicluster.*

Property 2.2 (Monotonicity). *Let (I, J) be a maximal bicluster. Any submatrix (I', J') , where $I' \supseteq I$ and $J' \supseteq J$, is not a bicluster.*

Usually, the efficient algorithms for enumerating CTV biclusters of ones from binary data are based on the monotonicity and anti-monotonicity properties [12]. In fact, we do not know any efficient algorithm for this task that is not based on these properties. RIn-Close (see Section 5), in turn, also considers these properties, but now in the context of numerical datasets. Any definition of a bicluster type that meets these properties can be used in our framework. For instance, our definition of a CVC bicluster considers the same maximum residue ϵ for all attributes, however it is possible to use a different maximum residue for each attribute. If we are analyzing a dataset where the attributes have different range values, we could, for example, use the percentage of the variation of each attribute as a maximum residue for each attribute. But, we also could normalize / scale the domain of the attributes and use the same maximum residue for all of them.

Property 2.3. *Let \mathfrak{B}_ϵ be an enumerative bicluster solution with maximum perturbation ϵ , and $\mathfrak{B}_{\epsilon'}$ be an enumerative bicluster solution with maximum perturbation ϵ' , where $\epsilon > \epsilon'$. Both, \mathfrak{B}_ϵ and $\mathfrak{B}_{\epsilon'}$, with the same restrictions in the minimum number of rows and columns of the biclusters. In this case, $\text{span}(\mathfrak{B}_\epsilon) \supseteq \text{span}(\mathfrak{B}_{\epsilon'})$, and therefore $\text{cov}(\mathfrak{B}_\epsilon) \geq \text{cov}(\mathfrak{B}_{\epsilon'})$.*

Property 2.3 states that the coverage is a monotonic function with respect to ϵ . However, it does not indicate that the number of biclusters will always increase with ϵ . With an increasing in ϵ , new biclusters tend to be found, but it is not a rule. For example, two biclusters found with $\epsilon = x$ can be merged into one with some $\epsilon > x$. In fact, if ϵ is too high, the entire dataset will be considered a single valid bicluster.

3. Formal Concept Analysis

Formal Concept Analysis (FCA) is a field of applied mathematics based on mathematical order theory, in particular on the theory of complete lattices [28]. Here, we will explain the basic principles of FCA. For more details refer to Ganter et al. [28].

Definition 3.1 (Formal Context). *A formal context is a triple (G, M, I) of two sets G and M , and a relation $I \subseteq G \times M$. Each $g \in G$ is interpreted as an object, and each $m \in M$ is interpreted as an attribute. In order to express that an object g is in a relation I with an attribute m , we write $(g, m) \in I$ or gIm . We read it as “the object g has the attribute m ”.*

Notice that a formal context can be easily represented by a binary matrix \mathbf{D} , where rows represent objects, and columns represent attributes. We will have $d_{gm} = 1$ if the object g has the attribute m , and we will have $d_{gm} = 0$ otherwise.

For a subset $A \subseteq G$ of objects, we define:

$$A' = \{m \in M | \forall g \in A : gIm\} \quad (7)$$

(the set of attributes common to the objects in A). Similarly, for a subset $B \subseteq M$, we define:

$$B' = \{g \in G \mid \forall m \in B : gIm\} \quad (8)$$

(the set of objects common to the attributes in B).

Definition 3.2 (Formal Concept). *A formal concept of the formal context (G, M, I) is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A' = B$, and $B' = A$. The subset A of a formal concept (A, B) is called *extent*, and the subset B is called *intent*.*

By the definition, we see that though many subsets A can generate the same subset B , only the largest (closed) subset A is part of a formal concept (and vice versa). Formal concepts are partially ordered by $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 (\Leftrightarrow B_2 \subseteq B_1)$. With respect to this partial order, the set of all formal concepts forms a complete lattice called the *concept lattice* of the formal context (G, M, I) , denoted by $\mathfrak{B}(G, M, I)$. There are several algorithms in the literature that are able to extract the concept lattice of a formal context. Some examples are: Close-by-One (CbO) [48], In-Close [6], In-Close2 [7], and FCbO [45]. As the algorithms that we will propose are based on In-Close2, Subsection 3.1 is devoted to its formalization.

3.1. In-Close2

In-Close2 [7] and its precursor In-Close [6] are based conceptually on Close-By-One [48]. These algorithms use the lexicographic approach for mining formal concepts, thus avoiding the discovery of the same formal concept multiple times. Ganter [26] showed how the lexicographical order of concepts can be used to avoid the search of repeated results. In the mathematical order theory, combinations have a lexicographical order, for instance, $\{1, 2, 3\}$ comes before $\{1, 2, 4\}$, and also before $\{1, 3\}$ [6]. In-Close and In-Close2 maintain a *current attribute*. The concept next generated is new (*canonical*) if its intent contains no attribute preceding the current attribute. Therefore, to verify canonicity, In-Close/In-Close2 does the following: supposing that B is the current intent, j is the current attribute, and RW is the resulting extent, RW is not canonical if

$$\exists k \in M \setminus B \mid [k < j] \wedge [\forall g \in RW : gIk]. \quad (9)$$

i.e., if there is an attribute $k < j$ where $k \notin B$ and $RW \subseteq \{k\}'$. See Eq. (8) for a definition of $\{k\}'$. The concept of canonicity was introduced in Kuznetsov [47].

Algorithm 1 shows In-Close2 pseudocode. When we use A_r and B_r , it means the extent and the intent of the r -th formal concept, respectively. When we write J_k it means the element of the set J at position k , for instance, if $J = \{2, 5, 7, 8, 13\}$ and $k = 2$, $J_2 = 5$. The same for R_k . In the main function of In-Close2, we set $(A_1, B_1) \leftarrow (\{1, \dots, n\}, \{\})$ (which is called *supremum*), and $r_{new} \leftarrow 1$.

Then, we call the function In-Close2 to incrementally close the formal concept (A_1, B_1) , beginning at attribute index 1. Thereafter, all formal concepts will be found recursively. During the closure of a formal concept, In-Close2 iterates across the attributes. If the current attribute j is not an inherited attribute, In-Close2 computes the candidate to a new extent RW . If the extent RW is the same as the current extent A_r , then attribute j is added to the current intent B_r . If the extent RW is not the same as the current extent A_r , In-Close2 tests if RW is canonical. If yes, the current formal concept (A_r, B_r) will give rise to a child formal concept. After the closure of the current formal concept (A_r, B_r) , In-Close2 starts to close its children.

Algorithm 1 In-Close2

Input: Binary data matrix $\mathbf{D}_{n \times m}$, minimum number of rows $minRow$, index of the formal concept to be closed r , index of the initial attribute y

```

1:  $J \leftarrow \{\}$ 
2:  $R \leftarrow \{\}$ 
3: for  $j \leftarrow y$  to  $m$  do
4:   if  $j \notin B_r$  then
5:      $RW \leftarrow A_r \cap \{j\}'$ 
6:     if  $|RW| \geq minRow$  then
7:       if  $|RW| = |A_r|$  then
8:          $B_r \leftarrow B_r \cup \{j\}$ 
9:       else if  $RW$  is canonical then
10:         $r_{new} \leftarrow r_{new} + 1$  // global variable
11:         $J \leftarrow J \cup \{j\}$ 
12:         $R \leftarrow R \cup \{r_{new}\}$ 
13:         $A_{r_{new}} \leftarrow RW$ 
14:   for  $k \leftarrow 1$  to  $|J|$  do
15:      $B_{R_k} \leftarrow B_r \cup \{J_k\}$ 
16:   In-Close2( $\mathbf{D}, minRow, R_k, J_k + 1$ )

```

The worst-case time of In-Close2 is $O(knm^2)$, where k is the number of biclusters. If $minRow = 1$, In-Close2 mines the concept lattice of the formal context represented by the binary matrix \mathbf{D} . Otherwise, if $minRow > 1$, In-Close2 mines the set of all *frequent concepts* for the threshold $minRow$, called the *iceberg concept lattice* [50]. In addition to the minimum number of rows $minRow$, we can easily add a minimum number of columns $minCol$ to In-Close2. While In-Close2 loops through the attributes, a formal concept (A_r, B_r) can be discarded if, even adding all remaining attributes to its intent, it will not meet the minimum number of columns $minCol$ (therefore, its next descendants will not meet the minimum number of columns $minCol$ as well). Although this restriction can be checked only during the closure of a formal concept, it will save computational resources because it avoids generating descendants that do not meet the restriction $minCol$.

3.2. FCA and related areas of research in the literature

The problem of extracting the concept lattice from a formal context is the same as extracting all maximal CTV biclusters of ones from a binary data matrix. A formal concept is a maximal CTV

bicluster of ones. Extent A and intent B are the set of rows (objects) and columns (attributes) that compose a bicluster, respectively.

The association mining problem is also closely related to FCA. This problem is divided in two sub-problems: (i) the frequent itemset (pattern) mining problem, and (ii) the problem of mining the association rules from these itemsets. As the first sub-problem is the most computationally expensive, almost all researches have been focused on the frequent itemset generation phase. In terms of FCA, the problem of mining all *frequent itemsets* (patterns) can be described as follows. Given a formal context (G, M, I) , determine all subsets $B \subseteq M$ such that the support of B ($\text{supp}(B) = |B'|$) is above a user-defined parameter [49]. Examples of algorithms that perform this task are Apriori [3] and Eclat [77]. To reduce the computational cost of the frequent pattern mining problem, some algorithms, such as GenMax [32], mine only the *maximal frequent itemsets*, i.e., those frequent itemsets from which all supersets are infrequent and all subsets are frequent. The problem of this approach is that it leads to a loss of information since the supports of the subsets are not available. An option to reduce the computational cost without loss of information is to mine only the *closed frequent itemsets*. A frequent itemset B is called closed if there exists no superset $D \supset B$ with $B' = D'$. The closed frequent itemsets are also called *frequent concept intents*. For any itemset B , its concept intent is given by B'' . Note that this approach is the most closely related to FCA. Remarkably, a concept lattice contains all necessary information to derive the support of all (frequent) itemsets [49]. Indeed, the set of closed frequent itemsets uniquely determines the exact frequency of all itemsets, and it can be orders of magnitude smaller than the set of all frequent itemsets [76]. Moreover, this approach drastically reduces the number of rules that have to be presented to the user, without any information loss [49]. CHARM [76] is a well-known algorithm to mine all closed frequent itemsets. It exploits the fact that the extents of the formal concepts are irrelevant in the frequent pattern mining problem (just the intents and the cardinality of the extents are relevant). Thus, it drastically cuts down the size of memory required [76].

The problem of enumerating all maximal bicliques from a bipartite graph is also closely related to FCA. Madeira and Oliveira [56] stated that in the simplest case of biclustering, where we are looking for CTV biclusters of ones in a binary data matrix \mathbf{D} , a bicluster corresponds to a biclique in the corresponding bipartite graph. Rows and columns of the matrix \mathbf{D} correspond, respectively, to the first and second sets of vertices of a bipartite graph. Each element d_{ij} is equal to 1 if vertex i is connected to vertex j , and 0 otherwise. Thus, the binary matrix \mathbf{D} is the adjacency matrix of a bipartite graph. In this scenario, a formal concept from the binary matrix \mathbf{D} is equivalent to a maximal biclique (bicluster). So, finding a concept lattice is also equivalent to finding all maximal bicliques of a bipartite graph. The connection between FCA and the problem of enumerating all maximal bicliques from a bipartite graph is explored in several papers [1, 29, 30]. Moreover, Gély *et al.* [30] pointed out

several algorithms to find all maximal bicliques from a bipartite graph, most of them are from the area of FCA.

4. Related Works

Due to the inherent computational complexity of the problem of finding all maximal biclusters, most of the proposed algorithms are heuristic-based [56]. Some relevant heuristics in the literature are: CC [16], FLOC [74], ROCC [23], ISA [37], Plaid [50], and OPSM [11]. CC looks for biclusters with mean squared residue (MSR) below a user-defined threshold δ . It mines one bicluster at each iteration, and performs random perturbations to the data to mask the already discovered bicluster. FLOC is also based on the MSR, but performs simultaneous bicluster identification. Briefly, the goal of CC and FLOC is to mine a set of biclusters with high average volume given the residue limit δ . ROCC is scalable and very versatile because it can be parametrized to mine several types of biclusters. It works in two steps: (i) find $k \times l$ biclusters arranged in a grid structure, keeping only the sr rows and sc columns with the lowest error associated with them, and (ii) filter out the biclusters with the largest error values, and merge similar biclusters. ISA looks for biclusters where their rows have an average value above a threshold t_g , and their columns have an average value above a threshold t_c . ISA starts with $nseed$ biclusters, and iteratively updates the columns and rows of the biclusters until convergence. Plaid fits parameters of a generative model of the data iteratively by minimizing the mean squared error between the modeled data and the true data. OPSM is a deterministic greedy algorithm dedicated to find large order-preserving submatrices.

Clearly, even the best heuristics potentially lead to sub-optimal solutions, so there are many proposals of exhaustive bicluster enumeration. Most of the work in this area is designed to mine all maximal CTV biclusters of ones from a binary dataset. In FCA, FPM and graph theory, there are several efficient algorithms able to perform this task. Proposals, such as [35, 57, 59, 61, 62, 67], binarize the dataset and use FCA, FPM or graph theory algorithms to enumerate the biclusters. To avoid the loss of information of tedious Boolean property encoding phases [12], many proposals deal directly with numerical datasets, as the following.

The proposals of [8, 12, 42, 43] are dedicated to enumerate CTV biclusters from numerical datasets.

The RCB algorithm [8] is based on an FPM algorithm called Apriori [4], which has a worst-case time exponential on the number of attributes. Thus, Apriori and the algorithms based on it are not efficient. It is also noteworthy that Apriori mines frequent itemsets, not closed frequent itemsets. Thus, it produces many redundant biclusters. RCB adopts a two step process. First, all the square submatrices that qualify to be a CTV bicluster are enumerated. Second, these square CTV biclusters are merged to form rectangular CTV biclusters of arbitrary sizes.

The NBS-Miner algorithm [12] mines all maximal CTV biclusters of a numerical dataset. The algorithm starts with the lattice $((\{\}, \{\}), (G, M))$ (whose bottom is $(\{\}, \{\})$ and top is (G, M)), i.e., the lattice containing all possible biclusters. Then, NBS-Miner explores its sublattices using three functions: enumeration, pruning, and propagation. The enumeration function splits recursively the current sublattice into two new sublattices. The prune function is responsible for pruning the sublattices that do not attend to the restriction of similarity (imposed by ϵ , see Eq. 1) or maximality. The propagation function implements the reduction of the size of the search space of a sublattice, not considering the entire current sublattice. The algorithm finds a bicluster when it finds a sublattice whose top is equal to the bottom.

Kaytue *et al.* [43] proposed two FCA-based methods to enumerate CTV biclusters. The former is based on the discretization of the numerical data matrix using *conceptual scaling* [28]. Let W be the set of values that an object $g \in G$ can take for an attribute $m \in M$. First of all, they compute all *tolerance classes* [40] from W . Then, they create one formal context for each class of tolerance and use FCA standard algorithms to enumerate the formal concepts from them. Each formal concept corresponds to a maximal CTV bicluster. The formal contexts are created in a way that avoids finding redundant CTV biclusters, but at a price of not finding some biclusters. Since the resulting binary tables may be numerous depending on the number of elements of W and the parameter ϵ , this method is not feasible in many real-world scenarios. The second method is divided in two phases. In the first one, it enumerates all the CVC biclusters using *interval pattern structures* (IPS) [27]. It is noteworthy that this method returns redundant CVC biclusters. In the second phase, CTV biclusters are extracted from the CVC biclusters, but this process is not so clear, because a CVC bicluster can give rise to many CTV biclusters.

In [42], the authors also use tolerance classes over the set of numbers W , and create one formal context for each class of tolerance. But they proposed a new algorithm called TriMax to mine the CTV biclusters from these formal contexts. TriMax is able to perform a complete, correct and non-redundant enumeration of all maximal CTV biclusters in a numerical dataset. But due to the scaling process, TriMax may be not feasible in many real-world scenarios too.

In the next two subsections, we highlight the competitors of the algorithms that we are proposing.

4.1. Enumerating CVC (or CVR) biclusters

RAP [64] is also based on Apriori [4]. The authors did not describe their strategy to avoid redundancy, but we conjecture that the best that can be done is a pairwise comparison of biclusters with k and $k + 1$ columns.

In [18, 19], the authors proposed a method based on *partition pattern structure* (PPS) [9]. Their proposal is not able to perform a complete enumeration because the components of the partition of

Table 1: Comparison between RIn-Close.CVC_P, RIn-Close.CVC and their competitors.

	Complete	Correct	Non-Redundant	Efficient
RIn-Close.CVC_P	✓	✓	✓	✓
RIn-Close.CVC	✓	✓	✓	✓
RAP	✓	✓		
PPS	◦	✓	✓	
IPS	✓	✓		
TCA	✓	✓		

¹The symbol ✓ indicates that the algorithm has the property. The symbol ◦ indicates that the authors claim their algorithm has the property, but it fails to exhibit the property.

the set G must be disjunct. They proposed a strategy based on a lattice to remove the redundancy, which is much faster than to compare one bicluster against all the others. But it is necessary to mine the redundant biclusters to make this verification, so it is not an efficient method.

In [41], the authors revisited the proposals of mining CVC biclusters using IPS [43] and PPS [18, 19]. They also proposed an approach based on Triadic Concept Analysis (TCA) [52]. From a numerical dataset, they derivate a triadic context given by (M, G, G, Y) such that $(m, g_1, g_2) \in Y$ iif $|d_{g_1 m} - d_{g_2 m}| \leq \epsilon$. Then, they use standard TCA algorithms to enumerate the triadic concepts, but not all triadic concepts are maximal CVC biclusters.

Table 1 shows a comparison between these proposals and the algorithms that we are proposing to enumerate CVC biclusters in Section 5. As we see, our proposals are the only algorithms that are able to perform an efficient, complete, correct and non-redundant enumeration of all maximal CVC biclusters.

4.2. Enumerating CHV biclusters

pCluster [73] was the first deterministic algorithm with an enumerative approach to mine CHV biclusters. pCluster computes all row-maximal biclusters with two columns and all column-maximal biclusters with two rows, prunes the unpromising biclusters, and stores the remaining column-maximal biclusters in a prefix tree. Then, pCluster makes a depth-first search in this prefix tree in order to mine larger biclusters. However, pCluster has several shortcomings. It does not find all biclusters, can find biclusters that do not attend the user-defined measure of similarity, and returns redundant biclusters. Furthermore, pCluster’s computational complexity is exponential w.r.t. the number of attributes.

Maple [66] is an improved version of pCluster and it is closer to an actual enumerative algorithm. It returns only non-redundant biclusters, but it does not have an efficient approach to do this: for each possible new bicluster, Maple must look at all previously generated biclusters to avoid redundancy. Besides, there are two scenarios where Maple fails in performing a complete and correct enumeration of all maximal biclusters. If two biclusters have the same set of objects and share some attributes, Maple

would return only one bicluster containing both of them (thus violating the user-defined measure of similarity). Maple also may miss biclusters due to its routine of pruning unpromising biclusters: Maple keeps an attribute-list ordered by some criterion. If a bicluster has a subset of objects and a superset of attributes of another bicluster, and its extra attributes are subsequently considering Maple’s attribute-list, Maple would prune it incorrectly. The worst-case time of Maple’s search is also exponential w.r.t. the number of attributes.

MicroCluster [78] constructs a multigraph that represents all row-maximal biclusters with two columns, where nodes represent attributes and edges represent sets of objects. It uses a depth-first search on the multigraph to mine the biclusters. We tested the authors’ MicroCluster implementation (<http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/Software>), and we observed that MicroCluster can fail in enumerating all the maximal biclusters and can return biclusters that violate the user-defined measure of similarity. Its worst-case time is also exponential w.r.t. the number of attributes. As Maple, MicroCluster must look at all previously generated biclusters to avoid redundancy, which is always a strategy to be avoided. After mining a CHV bicluster, MicroCluster verifies if the variation in its rows and columns attends user-defined measures of similarity too. If a bicluster does not attend these restrictions, it is discarded. Note that this verification can be done in any biclustering solution.

To reduce computational cost, algorithms such as SeqClus [72] and CPT [33] relaxed the definition of CHV biclusters. Instead of looking for every pair of attributes (or objects), they use a pivot attribute to compute the CHV biclusters. Thus, given a user-defined parameter ϵ , their biclusters will have residue less or equal to 2ϵ [33]. The biclustering solution of these algorithms depends on the choices of the pivot attributes. So, we can say that this strategy yields an approximate result for the actual enumeration. A genuine complete and correct enumeration of CHV biclusters would provide the same solution when dealing with the original matrix or with its transpose version. The CHV biclusters are fully preserved when rows become columns and columns become rows of the matrix. But unfortunately, when resorting to the pivot attribute, those techniques are prone to lose this fundamental property. Accordingly, these algorithms are not able to perform a complete and correct enumeration. As SeqClus and CPT are based on algorithms to mine frequent itemset, not closed frequent itemsets, they return redundant biclusters and are not efficient to mine maximal biclusters. So, this idea of mining CHV biclusters using a pivot attribute can be better explored. In fact, we could use this idea in RIn-Close_CHV (as we used in RIn-Close_CHV_P since, in this case, if a new candidate attribute is coherent with any attribute of the current bicluster, it will be coherent with all other attributes), which would lead to an efficient algorithm. But our main goal is to provide an algorithm able to perform a complete and correct enumeration of all maximal CHV biclusters. Moreover, despite not having polynomial delay, RIn-Close_CHV has good computational performance as we can see with the experimental results in

Table 2: Comparison between RIn-Close.CHV_P, RIn-Close.CHV and their competitors.

	Complete	Correct	Non-Redundant	Efficient
RIn-Close.CHV_P	✓	✓	✓	✓
RIn-Close.CHV	✓	✓	✓	
pCluster	○	○		
Maple	○	○	✓	
MicroCluster	○	○	✓	
SeqClus	○	○		
CPT	○	○		

¹The symbol ✓ indicates that the algorithm has the property. The symbol ○ indicates that the authors claim their algorithm has the property, but it fails to exhibit the property.

Section 6.

Table 2 shows a comparison between these proposals and the algorithms that we are proposing to enumerate CHV biclusters in Section 5. As we see, we are proposing algorithms with a number of additional features that are able to support a wider range of application scenarios, including the identification of biological indicators [54] and classification based on associations [53]. Moreover, a bicluster solution provided by these competitors is contained in the RIn-Close’s solution, given the adequate parameter ϵ . For instance, if a user parameterizes CPT with $\epsilon = 1$, it would obtain a biclustering solution with biclusters with residue up to 2. Parameterizing RIn-Close with $\epsilon = 2$, a user would obtain a biclustering solution containing all the maximal versions of the biclusters of the CPT’s solution, and possibly containing additional biclusters.

5. RIn-Close

In-Close2 has been specifically designed to extract all maximal CTV biclusters of ones from a binary data matrix. Now, we will propose generalizations of In-Close2 to enumerate other types of biclusters from numerical (not only binary, but also integer or real-valued) matrices. We call our family of algorithms RIn-Close.

We chose to adapt In-Close2 because: (i) it is easy to understand; (ii) it is one of the fastest algorithms of FCA; (iii) it has support to the desired minimum number of rows in a bicluster (the parameter *minRow*); (iv) it is easy to incorporate a support to the desired minimum number of columns in a bicluster (the parameter *minCol*); and (v) In-Close2 starts with all objects in the extent of a formal concept. This latter aspect of In-Close2 is very important when working with integer or real-valued data matrices. For instance, when finding CVC biclusters, given the current attribute, we can look for the subsets of rows of the extent that accomplish the similarity restriction ϵ .

5.1. Finding biclusters with constant values on columns (CVC)

The algorithms of this section compute an efficient, complete, correct and non-redundant enumeration of all maximal CVC biclusters. First, we will show how to extract perfect biclusters, because it is the easiest case. After that, given a user-defined parameter ϵ ($\epsilon > 0$), we will show how to extract non-perfect CVC biclusters with maximum residue ϵ , as presented in Eq. (2).

5.1.1. Perfect Biclusters

The adaptation of In-Close2 to enumerate all maximal perfect CVC biclusters, called RIn-Close_CVC_P, is straightforward. We have only one major difference. In In-Close2, each bicluster (A_r, B_r) can generate just one descendant per attribute, whereas in RIn-Close_CVC_P, each bicluster (A_r, B_r) can generate multiple descendants per attribute. It happens because In-Close2 just looks for blocks of 1s, whereas RIn-Close_CVC_P looks for any blocks of constant values on columns. Fig. 1 illustrates this difference. In Fig. 1(a), In-Close2 is closing the bicluster $(A_r = \{g_2, g_3, g_4, g_8, g_9, g_{10}, g_{11}, g_{15}\}, B_r = \{m_1, m_3, m_7\})$. When it tries to add the attribute m_8 , bicluster (A_r, B_r) gives rise to a new bicluster $(A = \{g_3, g_4, g_9, g_{15}\}, B = \{m_1, m_3, m_7, m_8\})$. In Fig. 1(b), RIn-Close_CVC_P is closing the same bicluster (A_r, B_r) , but when it tries to add the attribute m_8 , bicluster (A_r, B_r) gives rise to four new perfect CVC biclusters *without overlap between their extents*: (a) $(A = \{g_{11}\}, B = \{m_1, m_3, m_7, m_8\})$, (b) $(A = \{g_2, g_{15}\}, B = \{m_1, m_3, m_7, m_8\})$, (c) $(A = \{g_8, g_9, g_{10}\}, B = \{m_1, m_3, m_7, m_8\})$, and (d) $(A = \{g_3, g_4\}, B = \{m_1, m_3, m_7, m_8\})$. Note that the elements of the current attribute, m_8 , were sorted in order to easily identify all subsets of objects with constant values.

Algorithm 2 shows the pseudocode of RIn-Close_CVC_P. Notice that it is almost the same as In-Close2. There are basically two differences. The first one is that the current attribute j is added to the current intent B_r if all values of attribute j and objects A_r are equal. And the second one occurs by the fact that the bicluster (A_r, B_r) can generate multiple descendants. So, RIn-Close_CVC_P computes all the possible extents and loops across them. The test of canonicity is also essentially the same as in In-Close2. Let B be the current intent, j be the current attribute, and RW be the extent of the new bicluster, it is not canonical if

$$\exists k \in M \setminus B[k < j] \wedge [\max_{i \in RW} (d_{ik}) - \min_{i \in RW} (d_{ik}) = 0], \quad (10)$$

i.e., if there is an attribute $k < j$ that we can add to the bicluster (RW, B) and it remains a valid perfect CVC bicluster. The worst-case time of RIn-Close_CVC_P is almost the same as In-Close2: $O(knm(\log n + m))$. The difference is due to the use of a sorting algorithm to compute the possible extents.

	<table style="border-collapse: collapse; width: 100%;"> <tr><th>m_1</th><th>m_3</th><th>m_7</th></tr> <tr><td>g_2</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>g_3</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>g_4</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>g_8</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>g_9</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>g_{10}</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>g_{11}</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>g_{15}</td><td>1</td><td>1</td><td>1</td></tr> </table>	m_1	m_3	m_7	g_2	1	1	1	g_3	1	1	1	g_4	1	1	1	g_8	1	1	1	g_9	1	1	1	g_{10}	1	1	1	g_{11}	1	1	1	g_{15}	1	1	1	+	<table style="border-collapse: collapse; width: 100%;"> <tr><th>m_8</th></tr> <tr><td>0</td></tr> <tr><td>1</td></tr> <tr><td>1</td></tr> <tr><td>0</td></tr> <tr><td>1</td></tr> <tr><td>0</td></tr> <tr><td>0</td></tr> <tr><td>1</td></tr> </table>	m_8	0	1	1	0	1	0	0	1	=	<table style="border-collapse: collapse; width: 100%;"> <tr><th>m_1</th><th>m_3</th><th>m_7</th><th>m_8</th></tr> <tr><td>g_3</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>g_4</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>g_9</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>g_{15}</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> </table>	m_1	m_3	m_7	m_8	g_3	1	1	1	1	g_4	1	1	1	1	g_9	1	1	1	1	g_{15}	1	1	1	1
m_1	m_3	m_7																																																																							
g_2	1	1	1																																																																						
g_3	1	1	1																																																																						
g_4	1	1	1																																																																						
g_8	1	1	1																																																																						
g_9	1	1	1																																																																						
g_{10}	1	1	1																																																																						
g_{11}	1	1	1																																																																						
g_{15}	1	1	1																																																																						
m_8																																																																									
0																																																																									
1																																																																									
1																																																																									
0																																																																									
1																																																																									
0																																																																									
0																																																																									
1																																																																									
m_1	m_3	m_7	m_8																																																																						
g_3	1	1	1	1																																																																					
g_4	1	1	1	1																																																																					
g_9	1	1	1	1																																																																					
g_{15}	1	1	1	1																																																																					

(a) Example of the generation of a single descendant by In-Close2.

	<table style="border-collapse: collapse; width: 100%;"> <tr><th>m_1</th><th>m_3</th><th>m_7</th></tr> <tr><td>g_2</td><td>3</td><td>1</td><td>4</td></tr> <tr><td>g_3</td><td>3</td><td>1</td><td>4</td></tr> <tr><td>g_4</td><td>3</td><td>1</td><td>4</td></tr> <tr><td>g_8</td><td>3</td><td>1</td><td>4</td></tr> <tr><td>g_9</td><td>3</td><td>1</td><td>4</td></tr> <tr><td>g_{10}</td><td>3</td><td>1</td><td>4</td></tr> <tr><td>g_{11}</td><td>3</td><td>1</td><td>4</td></tr> <tr><td>g_{15}</td><td>3</td><td>1</td><td>4</td></tr> </table>	m_1	m_3	m_7	g_2	3	1	4	g_3	3	1	4	g_4	3	1	4	g_8	3	1	4	g_9	3	1	4	g_{10}	3	1	4	g_{11}	3	1	4	g_{15}	3	1	4	+	<table style="border-collapse: collapse; width: 100%;"> <tr><th>m_8</th></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>4</td></tr> <tr><td>3</td></tr> <tr><td>3</td></tr> <tr><td>3</td></tr> <tr><td>1</td></tr> <tr><td>2</td></tr> </table>	m_8	2	4	4	3	3	3	1	2	=	<table style="border-collapse: collapse; width: 100%;"> <tr><th>m_1</th><th>m_3</th><th>m_7</th><th>m_8</th></tr> <tr><td>g_{11}</td><td>3</td><td>1</td><td>4</td><td>1</td></tr> <tr style="background-color: #cccccc;"><td>g_2</td><td>3</td><td>1</td><td>4</td><td>2</td></tr> <tr style="background-color: #cccccc;"><td>g_{15}</td><td>3</td><td>1</td><td>4</td><td>2</td></tr> <tr><td>g_8</td><td>3</td><td>1</td><td>4</td><td>3</td></tr> <tr><td>g_9</td><td>3</td><td>1</td><td>4</td><td>3</td></tr> <tr><td>g_{10}</td><td>3</td><td>1</td><td>4</td><td>3</td></tr> <tr style="background-color: #cccccc;"><td>g_3</td><td>3</td><td>1</td><td>4</td><td>4</td></tr> <tr style="background-color: #cccccc;"><td>g_4</td><td>3</td><td>1</td><td>4</td><td>4</td></tr> </table>	m_1	m_3	m_7	m_8	g_{11}	3	1	4	1	g_2	3	1	4	2	g_{15}	3	1	4	2	g_8	3	1	4	3	g_9	3	1	4	3	g_{10}	3	1	4	3	g_3	3	1	4	4	g_4	3	1	4	4
m_1	m_3	m_7																																																																																											
g_2	3	1	4																																																																																										
g_3	3	1	4																																																																																										
g_4	3	1	4																																																																																										
g_8	3	1	4																																																																																										
g_9	3	1	4																																																																																										
g_{10}	3	1	4																																																																																										
g_{11}	3	1	4																																																																																										
g_{15}	3	1	4																																																																																										
m_8																																																																																													
2																																																																																													
4																																																																																													
4																																																																																													
3																																																																																													
3																																																																																													
3																																																																																													
1																																																																																													
2																																																																																													
m_1	m_3	m_7	m_8																																																																																										
g_{11}	3	1	4	1																																																																																									
g_2	3	1	4	2																																																																																									
g_{15}	3	1	4	2																																																																																									
g_8	3	1	4	3																																																																																									
g_9	3	1	4	3																																																																																									
g_{10}	3	1	4	3																																																																																									
g_3	3	1	4	4																																																																																									
g_4	3	1	4	4																																																																																									

(b) Example of the generation of multiple descendants by RIn-Close_CVC_P.

	<table style="border-collapse: collapse; width: 100%;"> <tr><th>m_1</th><th>m_3</th><th>m_7</th></tr> <tr><td>g_2</td><td>1</td><td>2</td><td>1</td></tr> <tr><td>g_3</td><td>2</td><td>2</td><td>1</td></tr> <tr><td>g_4</td><td>2</td><td>1</td><td>1</td></tr> <tr><td>g_8</td><td>1</td><td>2</td><td>1</td></tr> <tr><td>g_9</td><td>2</td><td>1</td><td>1</td></tr> <tr><td>g_{10}</td><td>1</td><td>1</td><td>2</td></tr> <tr><td>g_{11}</td><td>1</td><td>1</td><td>2</td></tr> <tr><td>g_{15}</td><td>2</td><td>1</td><td>1</td></tr> <tr><td>g_{16}</td><td>2</td><td>1</td><td>1</td></tr> <tr><td>g_{19}</td><td>2</td><td>1</td><td>2</td></tr> <tr><td>g_{20}</td><td>1</td><td>2</td><td>2</td></tr> <tr><td>g_{22}</td><td>2</td><td>2</td><td>1</td></tr> <tr><td>g_{23}</td><td>2</td><td>2</td><td>2</td></tr> </table>	m_1	m_3	m_7	g_2	1	2	1	g_3	2	2	1	g_4	2	1	1	g_8	1	2	1	g_9	2	1	1	g_{10}	1	1	2	g_{11}	1	1	2	g_{15}	2	1	1	g_{16}	2	1	1	g_{19}	2	1	2	g_{20}	1	2	2	g_{22}	2	2	1	g_{23}	2	2	2	+	<table style="border-collapse: collapse; width: 100%;"> <tr><th>m_8</th></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>4</td></tr> <tr><td>3</td></tr> <tr><td>3</td></tr> <tr><td>3</td></tr> <tr><td>1</td></tr> <tr><td>3</td></tr> <tr><td>3</td></tr> <tr><td>4</td></tr> <tr><td>3</td></tr> <tr><td>4</td></tr> </table>	m_8	2	4	4	3	3	3	1	3	3	4	3	4	=	<table style="border-collapse: collapse; width: 100%;"> <tr><th>m_1</th><th>m_3</th><th>m_7</th><th>m_8</th></tr> <tr style="background-color: #cccccc;"><td>g_{11}</td><td>1</td><td>1</td><td>2</td><td>1</td></tr> <tr style="background-color: #cccccc;"><td>g_2</td><td>1</td><td>2</td><td>1</td><td>2</td></tr> <tr style="background-color: #cccccc;"><td>g_{15}</td><td>2</td><td>1</td><td>1</td><td>2</td></tr> <tr style="background-color: #cccccc;"><td>g_8</td><td>1</td><td>2</td><td>1</td><td>3</td></tr> <tr style="background-color: #cccccc;"><td>g_9</td><td>2</td><td>1</td><td>1</td><td>3</td></tr> <tr style="background-color: #cccccc;"><td>g_{10}</td><td>1</td><td>1</td><td>2</td><td>3</td></tr> <tr style="background-color: #cccccc;"><td>g_{16}</td><td>2</td><td>1</td><td>1</td><td>3</td></tr> <tr style="background-color: #cccccc;"><td>g_{19}</td><td>2</td><td>1</td><td>2</td><td>3</td></tr> <tr style="background-color: #cccccc;"><td>g_{22}</td><td>2</td><td>2</td><td>1</td><td>3</td></tr> <tr style="background-color: #cccccc;"><td>g_3</td><td>2</td><td>2</td><td>1</td><td>4</td></tr> <tr style="background-color: #cccccc;"><td>g_4</td><td>2</td><td>1</td><td>1</td><td>4</td></tr> <tr style="background-color: #cccccc;"><td>g_{20}</td><td>1</td><td>2</td><td>2</td><td>4</td></tr> <tr style="background-color: #cccccc;"><td>g_{23}</td><td>2</td><td>2</td><td>2</td><td>4</td></tr> </table>	m_1	m_3	m_7	m_8	g_{11}	1	1	2	1	g_2	1	2	1	2	g_{15}	2	1	1	2	g_8	1	2	1	3	g_9	2	1	1	3	g_{10}	1	1	2	3	g_{16}	2	1	1	3	g_{19}	2	1	2	3	g_{22}	2	2	1	3	g_3	2	2	1	4	g_4	2	1	1	4	g_{20}	1	2	2	4	g_{23}	2	2	2	4
m_1	m_3	m_7																																																																																																																																												
g_2	1	2	1																																																																																																																																											
g_3	2	2	1																																																																																																																																											
g_4	2	1	1																																																																																																																																											
g_8	1	2	1																																																																																																																																											
g_9	2	1	1																																																																																																																																											
g_{10}	1	1	2																																																																																																																																											
g_{11}	1	1	2																																																																																																																																											
g_{15}	2	1	1																																																																																																																																											
g_{16}	2	1	1																																																																																																																																											
g_{19}	2	1	2																																																																																																																																											
g_{20}	1	2	2																																																																																																																																											
g_{22}	2	2	1																																																																																																																																											
g_{23}	2	2	2																																																																																																																																											
m_8																																																																																																																																														
2																																																																																																																																														
4																																																																																																																																														
4																																																																																																																																														
3																																																																																																																																														
3																																																																																																																																														
3																																																																																																																																														
1																																																																																																																																														
3																																																																																																																																														
3																																																																																																																																														
4																																																																																																																																														
3																																																																																																																																														
4																																																																																																																																														
m_1	m_3	m_7	m_8																																																																																																																																											
g_{11}	1	1	2	1																																																																																																																																										
g_2	1	2	1	2																																																																																																																																										
g_{15}	2	1	1	2																																																																																																																																										
g_8	1	2	1	3																																																																																																																																										
g_9	2	1	1	3																																																																																																																																										
g_{10}	1	1	2	3																																																																																																																																										
g_{16}	2	1	1	3																																																																																																																																										
g_{19}	2	1	2	3																																																																																																																																										
g_{22}	2	2	1	3																																																																																																																																										
g_3	2	2	1	4																																																																																																																																										
g_4	2	1	1	4																																																																																																																																										
g_{20}	1	2	2	4																																																																																																																																										
g_{23}	2	2	2	4																																																																																																																																										

(c) Example of the generation of multiple descendants by RIn-Close_CVC (considering $\epsilon = 1$).

Figure 1: Generation of descendants by In-Close2, and RIn-Close_CVC_P and and RIn-Close_CVC.

Algorithm 2 RIn-Close_CVC_P

Input: Data matrix $\mathbf{D}_{n \times m}$, minimum number of rows $minRow$, index of the bicluster to be closed r , index of the initial attribute y

```
1:  $J \leftarrow \{\}$ 
2:  $R \leftarrow \{\}$ 
3: for  $j \leftarrow y$  to  $m$  do
4:   if  $j \notin B_r$  then
5:     if  $\max_{i \in A_r}(d_{ij}) - \min_{i \in A_r}(d_{ij}) = 0$  then
6:        $B_r \leftarrow B_r \cup \{j\}$ 
7:     else
8:       Compute the possible extents
9:       for each possible extent  $RW$  do
10:        if  $|RW| \geq minRow$  and  $RW$  is canonical then
11:           $r_{new} \leftarrow r_{new} + 1$ 
12:           $R \leftarrow R \cup \{r_{new}\}$ 
13:           $J \leftarrow J \cup \{j\}$ 
14:           $A_{r_{new}} \leftarrow RW$ 
15: for  $k \leftarrow 1$  to  $|J|$  do
16:    $B_{R_k} \leftarrow B_r \cup \{J_k\}$ 
17:   RIn-Close_CVC_P( $\mathbf{D}, minRow, R_k, J_k + 1$ )
```

5.1.2. Non-Perfect Biclusters

This adaptation of In-Close2, called RIn-Close_CVC, is significantly more elaborate than RIn-Close_CVC_P because, besides a bicluster (A_r, B_r) being able to generate multiple descendants per attribute, there may be overlaps between their extents. For instance, in Fig. 1(c), RIn-Close_CVC is closing the bicluster $(A_r = \{g_2, g_3, g_4, g_8, g_9, g_{10}, g_{11}, g_{15}, g_{16}, g_{19}, g_{20}, g_{22}, g_{23}\}, B_r = \{m_1, m_3, m_7\})$, when it tries to add the current attribute m_8 , bicluster (A_r, B_r) gives rise to three new biclusters *with overlap between their extents* (considering $\epsilon = 1$). Notice again that the elements of the current attribute were sorted to facilitate the identification of all possible extents.

Due to a bicluster (A_r, B_r) being able to generate multiple descendants per attribute, with overlap between them, it is necessary to take some actions to avoid the generation of duplicate and non-maximal biclusters. In fact, these challenging issues can occur if two descendant biclusters share $minRow$ rows or more in their extents.

Assuming $minRow = 3$, in our example in Fig. 1(c), biclusters (a) and (b) can not generate duplicate biclusters because they share only 2 rows in their extents. But biclusters (b) and (c) can generate duplicate biclusters with extent $A \subseteq \{g_8, g_9, g_{10}, g_{16}, g_{19}, g_{22}\}$ and $|A| \geq minRow$, when adding a new attribute. To solve this problem, we added one more verification on the test of canonicity. This new verification is based on the fact that two distinct CVC biclusters must have two distinct extents. So, we track the extents that have already been generated using efficient symbol table implementations, such as hash tables (HTs) or balanced search trees (BSTs). So, symbol table's keys are given by the extents, in such way that the rows in an extent are in their ascending (or descending) order. The worst-case time to insert and search in a BST is $O(\log k)$, where k is its number of elements. The

worst-case time to insert and search in a HT is $O(1)$ and $O(k)$, respectively. However, under reasonable assumptions, the average time to search in a HT is $O(1)$. The remainder of the test of canonicity is again essentially the same as in In-Close2. Supposing that B is the current intent, j is the current attribute, and RW is the extent of the new bicluster, it is not canonical if

$$\exists k \in M \setminus B [k < j] \wedge [\max_{i \in RW} (d_{ik}) - \min_{i \in RW} (d_{ik}) \leq \epsilon], \quad (11)$$

i.e., if there is an attribute $k < j$ that we can add to the bicluster (RW, B) and it remains a valid CVC bicluster.

But even with this new verification on the test of canonicity, we still have the undesirable possibility of generating non-maximal biclusters. For instance, in Fig. 1(c), bicluster (c) can give rise to the bicluster $(A = \{g_4, g_8, g_9, g_{10}\}, B = \{m_1, m_3, m_7, m_8, m_{11}, m_{16}\})$, and bicluster (b) can give rise to the bicluster $(A = \{g_8, g_9, g_{10}\}, B = \{m_1, m_3, m_7, m_8, m_{11}, m_{16}\})$, which is clearly non-maximal. So, when two biclusters share $minRow$ rows or more in their extents, we need to verify if their descendants are maximal in their extents (row-maximal). Therefore, the descendants of biclusters (b) and (c), whose extents are contained in the shared rows, need to check if they are row-maximal. Suppose that RM is the set of rows that the bicluster (A, B) must check to find out if it is row-maximal. The bicluster (A, B) is not row-maximal if there is an object $g \in RM$ that we can add to the bicluster and it remains a valid CVC bicluster, i.e.,

$$\exists g \in RM [\max_{i \in \{A \cup \{g\}\}} (d_{ij}) - \min_{i \in \{A \cup \{g\}\}} (d_{ij}) \leq \epsilon], \forall j \in B. \quad (12)$$

To explain how to compute RM , let us see the example in Fig. 2, which considers $\epsilon = 3$ and $minRow = 2$. Suppose that when adding attribute m_x , a bicluster generated four biclusters: (a), (b), (c) and (d), whose extents are highlighted in Fig. 2. Let us compute the set of rows RM that the descendants of the bicluster (b) must check to verify their maximality, i.e., $RM_{(b)}$. As the problem occurs when biclusters share $minRow$ rows or more in their extents, the pivot elements to compute $RM_{(b)}$ are g_e and g_h because they are the $minRow$ -th first and last element of (b), respectively. Their values are $g_e = 3$ and $g_h = 5$. Rows with values greater than or equal to 0 ($g_e - \epsilon$) or less than or equal to 8 ($g_h + \epsilon$) must comprise $RM_{(b)}$, so $RM_{(b)} = \{g_a, g_b, g_c, g_j\}$.

Back to our example in Fig. 1(c), $RM_{(b)} = \{g_3, g_4, g_{20}, g_{23}\}$. So, all descendants of the bicluster (b) with extent $A \subseteq \{g_8, g_9, g_{10}, g_{16}, g_{19}, g_{22}\}$ must test the rows in $RM_{(b)}$ to verify if they are row-maximal. For simplicity, the result will be correct if we implement this verification for all descendants of a bicluster.

It is very important to note that biclusters also need to inherit the set RM of their parents. For instance, suppose that the bicluster (b) of Fig. 1(c) gives rise to a bicluster $(A_x = \{g_8, g_9, g_{19}, g_{22}\}, B_x)$.

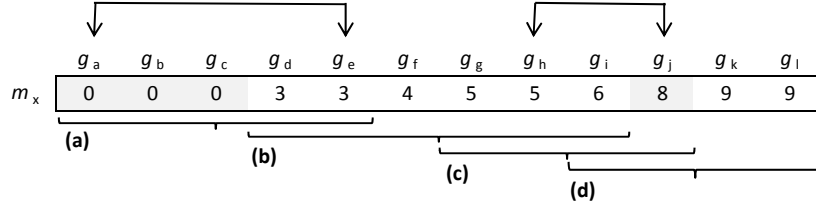


Figure 2: Example of how to find RM (considering $\epsilon = 3$ and $minRow = 2$).

So, we must have $RM_x \supseteq RM_{(b)}$ because the descendants of (A_x, B_x) must test the rows in $RM_{(b)}$ to verify if they are maximal.

Algorithm 3 RIn-Close_CVC

Input: Data matrix $\mathbf{D}_{n \times m}$, minimum number of rows $minRow$, index of the bicluster to be closed r , index of the initial attribute y , similarity constraint ϵ

```

1:  $J \leftarrow \{\}$ 
2:  $R \leftarrow \{\}$ 
3: for  $j \leftarrow y$  to  $m$  do
4:   if  $j \notin B_r$  then
5:     if  $\max_{i \in A_r}(d_{ij}) - \min_{i \in A_r}(d_{ij}) \leq \epsilon$  then
6:        $B_r \leftarrow B_r \cup \{j\}$ 
7:     else
8:       Compute the possible extents
9:       for each possible extent  $RW$  do
10:        if  $|RW| \geq minRow$  and  $RW$  is canonical and  $RW$  is row-maximal then
11:          Sort the elements of  $RW$ 
12:           $r_{new} \leftarrow r_{new} + 1$ 
13:           $R \leftarrow R \cup \{r_{new}\}$ 
14:           $J \leftarrow J \cup \{j\}$ 
15:           $A_{r_{new}} \leftarrow RW$ 
16:          Set  $RM_{r_{new}}$ 
17:          Update the symbol table
18: for  $k \leftarrow 1$  to  $|J|$  do
19:    $B_{R_k} \leftarrow B_r \cup \{J_k\}$ 
20:   RIn-Close_CVC( $\mathbf{D}, minRow, R_k, J_k + 1, \epsilon$ )

```

Algorithm 3 shows the pseudocode of RIn-Close_CVC. The worst-case time of RIn-Close_CVC is $O(kmn(mn + x))$, where x is the worst-case time of searching in the symbol table, so $x = O(\log k)$ for BSTs and $x = O(k)$ for HTs. But recall that HTs have a much better computational cost on average: $O(1)$.

5.2. Finding biclusters with coherent values (CHV)

Once again, we will first show how to enumerate perfect CHV biclusters. We named this algorithm RIn-Close_CHV_P. It is very similar to RIn-Close_CVC_P. Secondly, we will show how to enumerate non-perfect CHV biclusters. We named this algorithm RIn-Close_CHV.

5.2.1. Perfect Biclusters

When we are looking for CVC or CVR biclusters, we look directly to the values of the data matrix. But when we are looking for CHV biclusters, we need to check if there is coherence (additive or multiplicative) between each pair of columns (or rows) of the data matrix. For this, in RIn-Close_CHV_P, a bicluster starts with one column in its intent, which we call *pivot column*. Then, RIn-Close_CHV_P mines all biclusters that have this pivot column in their intents. Algorithm 4 shows this procedure. At the first iteration of the loop, RIn-Close_CHV_P will find all biclusters that have column 1 in their intents; at the second iteration, RIn-Close_CHV_P will find all biclusters that have column 2 and do not have column 1 in their intents; at the third iteration, RIn-Close_CHV_P will find all biclusters that have column 3 and do not have column 1 and 2 in their intents; and so on.

RIn-Close_CHV_P exploits the fact that for mining perfect CHV biclusters it is not necessary to check if there is coherence between all pair of columns in an intent. Note that in RIn-Close_CHV_P pseudocode, Algorithm 5, we just compute the difference between the current attribute j and the pivot column of the current intent B_r , i.e., B_{r_1} . If the current attribute j matches perfectly the pivot column, it will match perfectly the other columns of the intent B_r as well.

Algorithm 4 Main_RIn-Close_CHV_P

Input: Data matrix $\mathbf{D}_{n \times m}$, minimum number of rows $minRow$

- 1: $r_{new} \leftarrow 0$ // global variable
 - 2: **for** $atr \leftarrow 1$ to $m - 1$ **do**
 - 3: $r_{new} \leftarrow r_{new} + 1$
 - 4: $A_{r_{new}} \leftarrow \{1, \dots, n\}$
 - 5: $B_{r_{new}} \leftarrow \{atr\}$
 - 6: RIn-Close_CHV_P($\mathbf{D}, minRow, r_{new}, atr + 1$)
-

The test of canonicity is also essentially the same as in In-Close2. Let B be the current intent, j be the current attribute, and RW be the extent of the new bicluster. It is not canonical if

$$\exists k \in M \setminus B \mid [k < j] \wedge [\max(Z^{B_{r_1}k}) - \min(Z^{B_{r_1}k}) = 0], \quad (13)$$

where $Z^{B_{r_1}k} \leftarrow \{d_{iB_{r_1}} - d_{ik}\}_{i \in RW}$, i.e., if there is an attribute $k < j$ that we can add to the bicluster and it remains a valid perfect CHV bicluster.

The worst-case time of RIn-Close_CHV_P is the same as that of RIn-Close_CVC_P: $O(knm(\log n + m))$.

5.2.2. Non-Perfect Biclusters

Now, we will explain how to perform a complete, correct and non-redundant enumeration of all maximal perturbed CHV biclusters, given a similarity constraint determined by the user-defined parameter ϵ ($\epsilon > 0$), as presented in Eq. (3).

Algorithm 5 RIn-Close_CHV_P

Input: Data matrix $\mathbf{D}_{n \times m}$, minimum number of rows $minRow$, index of the bicluster to be closed r , index of the initial attribute y

```
1:  $J \leftarrow \{\}$ 
2:  $R \leftarrow \{\}$ 
3: for  $j \leftarrow y$  to  $m$  do
4:   if  $j \notin B_r$  then
5:      $Z \leftarrow \{d_{iB_{r_1}} - d_{ij}\}_{i \in A_r}$ 
6:     if  $\max(Z) - \min(Z) = 0$  then
7:        $B_r \leftarrow B_r \cup \{j\}$ 
8:     else
9:       Compute the possible extents
10:      for each possible extent  $RW$  do
11:        if  $|RW| \geq minRow$  and  $RW$  is canonical then
12:           $r_{new} \leftarrow r_{new} + 1$ 
13:           $R \leftarrow R \cup \{r_{new}\}$ 
14:           $J \leftarrow J \cup \{j\}$ 
15:           $A_{r_{new}} \leftarrow RW$ 
16: for  $k \leftarrow 1$  to  $|J|$  do
17:    $B_{R_k} \leftarrow B_r \cup \{J_k\}$ 
18:   RIn-Close_CHV_P( $\mathbf{D}, minRow, R_k, J_k + 1$ )
```

To achieve this goal, we can not simply apply to RIn-Close_CHV_P the same adaptations that we have made in RIn-Close_CVC_P to achieve RIn-Close_CVC. First of all, if RIn-Close_CHV computed the set Z considering only the pivot column B_{r_1} and the current attribute j , it could occur a difference up to 2ϵ between any other two columns of B_r . Besides, the order of choice of the pivot columns interfere in the outcome in this scenario. In this way RIn-Close_CHV would yield just an approximate result of an actual enumeration (as SeqClus [72] and CPT [33] do), and this is not the case here. Also, RIn-Close_CHV could not simply verify if the current attribute j fits all the attributes in the current intent B_r . An example of a problem that could happen is that: if the data matrix has two biclusters with the same extent A , but different intents $B_x = \{m_1, m_3, m_5\}$ and $B_y = \{m_1, m_3, m_6, m_8\}$, a naive procedure would find just the first one because it loops through the attributes in its sequential order and the attribute m_5 is not coherent with attributes m_6 and m_8 (considering the rows in extent A). Again, this is not the case here. In addition, it would be quite difficult to define the possible new extents. Another challenging issue of this approach is to determine when a bicluster is canonical or not. For instance, if the data matrix has two biclusters with the same extent A , but different intents $B_x = \{m_1, m_2, m_3, m_4\}$ and $B_y = \{m_2, m_3, m_4, m_5\}$, a naive procedure would discard the second because the attributes m_2, m_3 and m_4 are coherent with attribute m_1 . Not to mention that a non-canonical bicluster could give rise to a canonical bicluster.

To avoid all these undesired scenarios, RIn-Close_CHV uses the following procedure with three steps:

1. Compute the augmented matrix of the data matrix \mathbf{D} , denoted \mathbf{D}_a . \mathbf{D}_a is a matrix with the

difference between all pairs of columns of \mathbf{D} . For instance, the augmented matrix \mathbf{D}_a of the data matrix \mathbf{D} in Table 3 is illustrated in Table 4. The first column of \mathbf{D}_a is the difference between columns 1 and 2 of \mathbf{D} , the second column of \mathbf{D}_a is the difference between columns 1 and 3 of \mathbf{D} , the third column of \mathbf{D}_a is the difference between columns 1 and 4 of \mathbf{D} , and so on for all combinations of pairs of columns.

2. Apply RIn-Close_CVC to the augmented matrix \mathbf{D}_a . To illustrate, Fig. 3(a) shows all maximal CVC biclusters found by RIn-Close_CVC when applied to the data matrix of Table 4 (using $\text{minRow} = 2$ and $\epsilon = 1$).
3. Extract all maximal CHV biclusters from the maximal CVC biclusters found by RIn-Close_CVC (see the pseudocode in Algorithm 6).

Table 3: Example of a numerical dataset [12]

	m_1	m_2	m_3	m_4	m_5
g_1	1	2	2	1	6
g_2	2	1	1	0	6
g_3	2	2	1	7	6
g_4	8	9	2	6	7

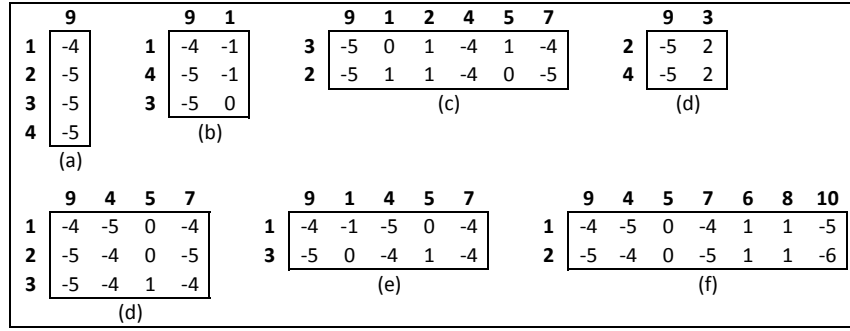
Table 4: Augmented matrix of the data matrix in Table 3.

	1	2	3	4	5	6	7	8	9	10
1	-1	-1	0	-5	0	1	-4	1	-4	-5
2	1	1	2	-4	0	1	-5	1	-5	-6
3	0	1	-5	-4	1	-5	-4	-6	-5	1
4	-1	6	2	1	7	3	2	-4	-5	-1

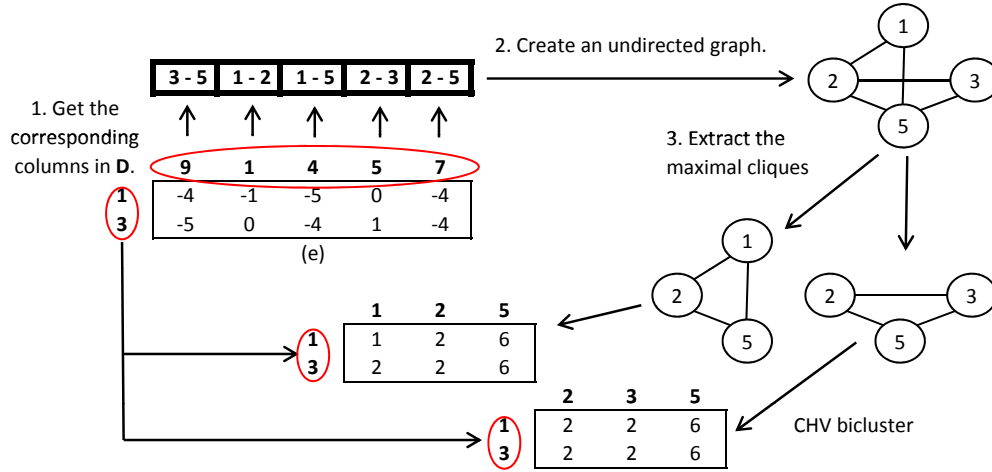
Algorithm 6 shows the pseudocode of the procedure to process each CVC bicluster. Steps in lines 2 to 5 are illustrated in Fig. 3(b). In this illustrative scheme, we are extracting CHV biclusters from the CVC bicluster (e) of Fig. 3(a). The first step is to get the corresponding columns in \mathbf{D} of the intent of bicluster (e). For instance, column 9 of \mathbf{D}_a corresponds to the difference between columns 3 and 5 of \mathbf{D} . Therefore, columns 3 and 5 are coherent with each other considering the extent $\{1, 3\}$ and $\epsilon = 1$. Let us name this corresponding set of columns as $B2$. The second step is to create an undirected graph, in which the nodes represent $B2$, and the edges represent the columns that are coherent with each other. The third step is to find all maximal cliques from this undirected graph. Each one of these cliques indicates the subsets of $B2$ in which all columns are coherent with each other considering the user-defined parameter ϵ . Thus, the CHV biclusters generated by the CVC bicluster (e) are: $(\{1, 3\}, \{1, 2, 5\})$ and $(\{1, 3\}, \{2, 3, 5\})$. Lines 7 and 8 of the pseudocode in Algorithm 6 verify if the CHV bicluster (C, D) is new and, if so, store it in the list of CHV biclusters. A CHV bicluster (C, D) is new if (i) its intent D is equal to $B2$, or (ii) (C, D) is row-maximal (there is no object g that we can add to its extent C), i.e.,

$$\nexists g \in G \setminus C \mid [\max(Z^{jl}) - \min(Z^{jl}) \leq \epsilon], \forall j, l \in D, \quad (14)$$

where $Z^{jl} = \{d_{ij} - d_{il}\}_{i \in C \cup \{g\}}$. The worst-case time of verifying if a CHV bicluster is row-maximal is $O(mn)$. Makino and Uno [58] proved that all maximal cliques of a v vertices graph can be enumerated



(a) All CVC biclusters found in the data matrix of Table 4 (using $\minRow = 2$ and $\epsilon = 1$).



(b) Illustrative scheme to show how a CVC bicluster is processed by RIn-Close_CHV.

Figure 3: RIn-Close_CHV's framework.

with time delay $O(M(v))$, where $M(v)$ is the cost of multiplying two $v \times v$ matrices. As stated in Subsection 5.1.2, each CVC bicluster can be enumerated with time delay $O(m_a n(m_a n + x))$, where $m_a = m(m-1)/2$ is the number of columns of the augmented matrix \mathbf{D}_a .

Algorithm 6 Mining CHV biclusters from CVC biclusters

Input: List of CVC biclusters $lbCVC$

Output: List of CHV biclusters

- 1: **for each** (A, B) in $lbCVC$ **do**
 - 2: Compute $B2$
 - 3: Create an undirected graph
 - 4: Find all maximal cliques
 - 5: Compute the CHV biclusters
 - 6: **for each** CHV bicluster (C, D) **do**
 - 7: **if** $D = B2$ **or** (C, D) is row-maximal **then**
 - 8: Store (C, D) in the list of CHV biclusters
-

6. Experimental Results

We evaluated the RIn-Close family of algorithms on both synthetic and real datasets. Our goals are to highlight various practicalities in the usage of RIn-Close, and to outline the advantages and distinct aspects of an enumerative algorithm when compared to heuristics.

We implemented RIn-Close using C++. For the third step of RIn-Close-CHV, we implemented the BK algorithm [13] with the I. Kochs pivot selection strategy [44], because it is the best one in practice [15]. For the heuristics, we used the MTBA toolbox [38]. The only exception was ROCC's implementation that was obtained from the authors. The experiments were carried out on a PC Intel(R) Core(TM) i7-4770K CPU @ 3.5 GHz, 32 GB of RAM, and running under Ubuntu 14.04. The codes of RIn-Close algorithms are available at <https://sourceforge.net/projects/rinclose/>.

6.1. RIn-Close's scalability

This experiment aims to test RIn-Close's performance when varying (i) the number n of rows of the dataset, (ii) the number m of columns of the dataset, (iii) the number of biclusters in the dataset, (iv) the bicluster row size, (v) the bicluster column size, (vi) the overlap, and (vii) the noise in the dataset. For this purpose, we created synthetic datasets that let us embed biclusters and then test how RIn-Close performs when varying each one of these parameters in isolation. The default parameters used in the synthetic data generator were: $n = 5000$, $m = 60$, number of biclusters = 10, bicluster row size = 200, bicluster column size = 8, overlap = 0.2, and Gaussian noise with $\mu = 0$ and $\sigma = 0.01$. The synthetic data generator creates the biclusters and assigns random values to the other regions of the dataset. Then, it adds the Gaussian noise and shuffles the rows and columns of the dataset. Therefore, the generator creates arbitrarily positioned overlapping biclusters, so that

the resulting biclusters are usually non-contiguous. For each configuration, we created 50 synthetic datasets to compute the average runtimes. RIn-Close_CVC_P and RIn-Close_CHV_P, that look for perfect biclusters, were applied to datasets without noise.

Figs. 4 to 7 show, respectively, the sensitivity to different datasets' configurations of RIn-Close_CVC_P, RIn-Close_CVC, RIn-Close_CHV_P, and RIn-Close_CHV. The first note on these results is that the runtime of Step 1 and 3 of RIn-Close_CHV was negligible. Therefore, the results presented in Fig. 7 are basically the runtime of the Step 2 of the algorithm, which is to mine the CVC biclusters from the augmented matrix \mathbf{D}_a .

The runtime increased linearly with n , except for RIn-Close_CHV, for which it increased logarithmically. But for all algorithms without exception, the runtime growth rate was better than their worst-case time complexities (see Section 5). For the variable m , the runtime increased linearly for RIn-Close_CVC_P and RIn-Close_CVC, which is better than their worst-case time complexities. For RIn-Close_CHV_P and RIn-Close_CHV, the runtime increased polynomially with m , which coincides with their worst-case time complexities. The runtime increased linearly with the number of biclusters, except for RIn-Close_CHV. For RIn-Close_CHV, we can notice a tendency of a logarithmically growth, which is good news because, due to its worst-case time complexity, we expected a linear growth too. For the bicluster row size, we had a linear growth of the runtime, except for RIn-Close_CHV, for which we had a smooth polynomial growth. RIn-Close_CVC_P and RIn-Close_CVC had the same behavior for the bicluster column size: the runtime increased logarithmically, but saturates and started to decrease linearly. With the increase in the bicluster column size, the coverage of the dataset increases, and it seemed to help these algorithms to find the biclusters more quickly. With more columns in the biclusters, more the inheritance of the columns tends to have a positive effect, and also less columns tend to be tested in the canonicity function. For the RIn-Close_CHV_P and RIn-Close_CHV, the runtime increased logarithmically and polynomially with the bicluster column size, respectively. For all algorithms, the runtime decreased linearly with the overlap. The noise did not greatly affect the runtime.

The variations presented in the boxplots are due to two main reasons: (i) an intrinsic variation due to the machine; and (ii) some characteristics of the synthetic datasets, more specifically, due to the arrangement of the biclusters and the noise in the datasets. The arrangement of the biclusters in the datasets impacts in the heritage of the columns, in the number of times that the canonicity function is called, and in the number of columns that are tested in the canonicity function. The noise in the datasets impacts in the computation of the possible extents, and also in the number of times that the canonicity function is called.

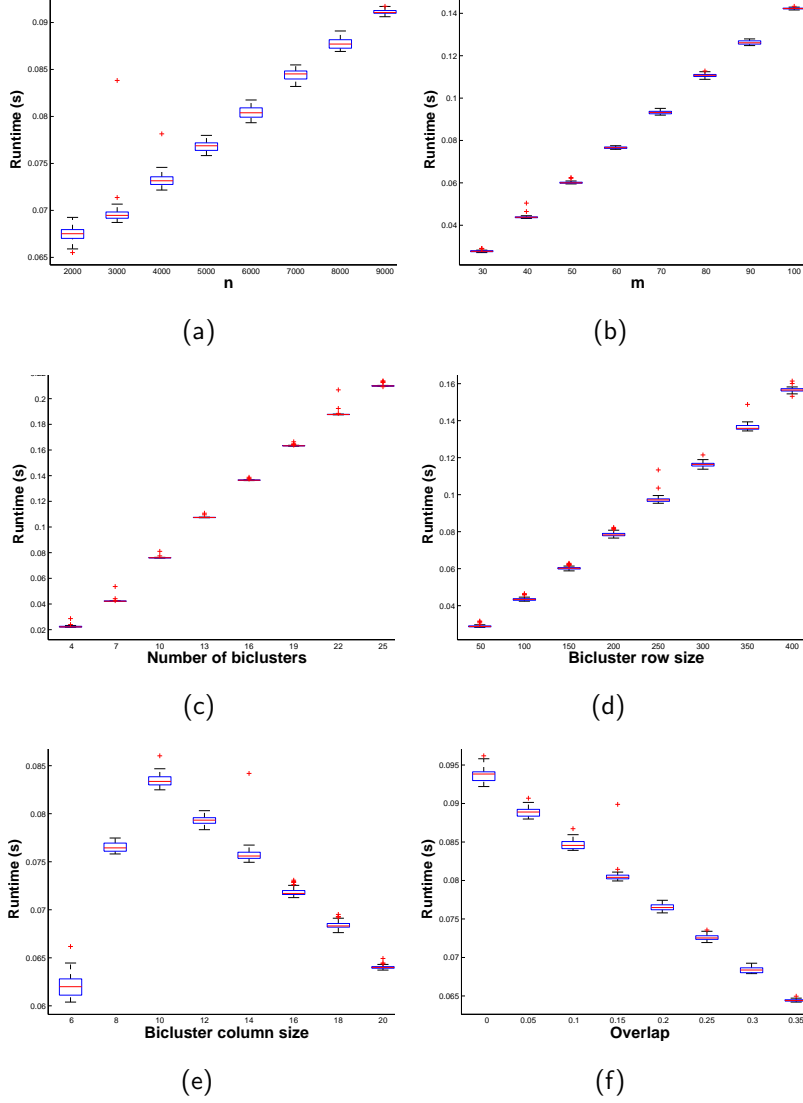


Figure 4: Results of the performance of RIn-Close.CVC.P when varying (a) the number n of rows of the dataset, (b) the number m of columns of the dataset, (c) the number of biclusters in the dataset, (d) the bicluster row size, (e) the bicluster column size, and (f) the overlap.

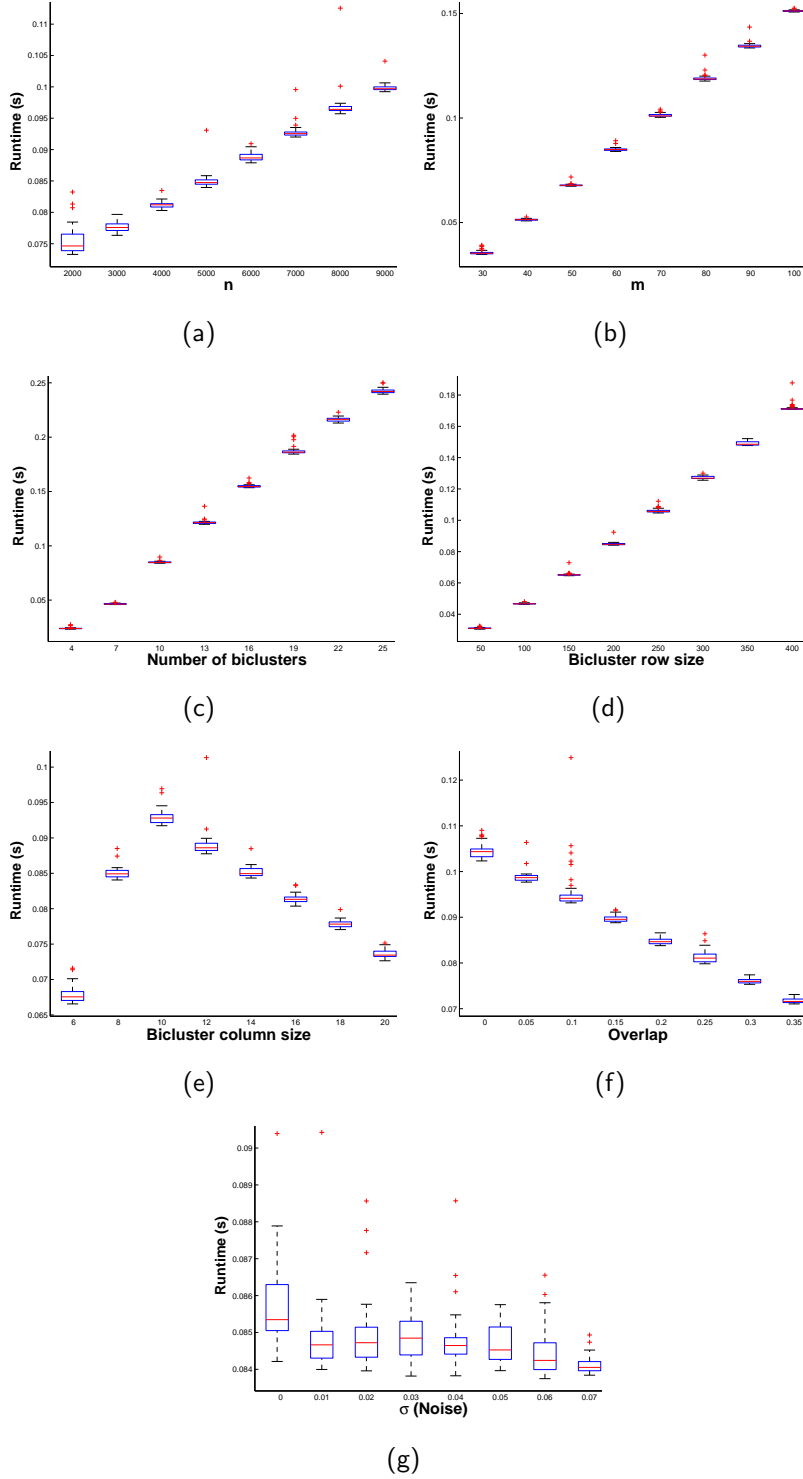


Figure 5: Results of the performance of RIn-Close_CVC when varying (a) the number n of rows of the dataset, (b) the number m of columns of the dataset, (c) the number of biclusters in the dataset, (d) the bicluster row size, (e) the bicluster column size, (f) the overlap, and (g) the noise.

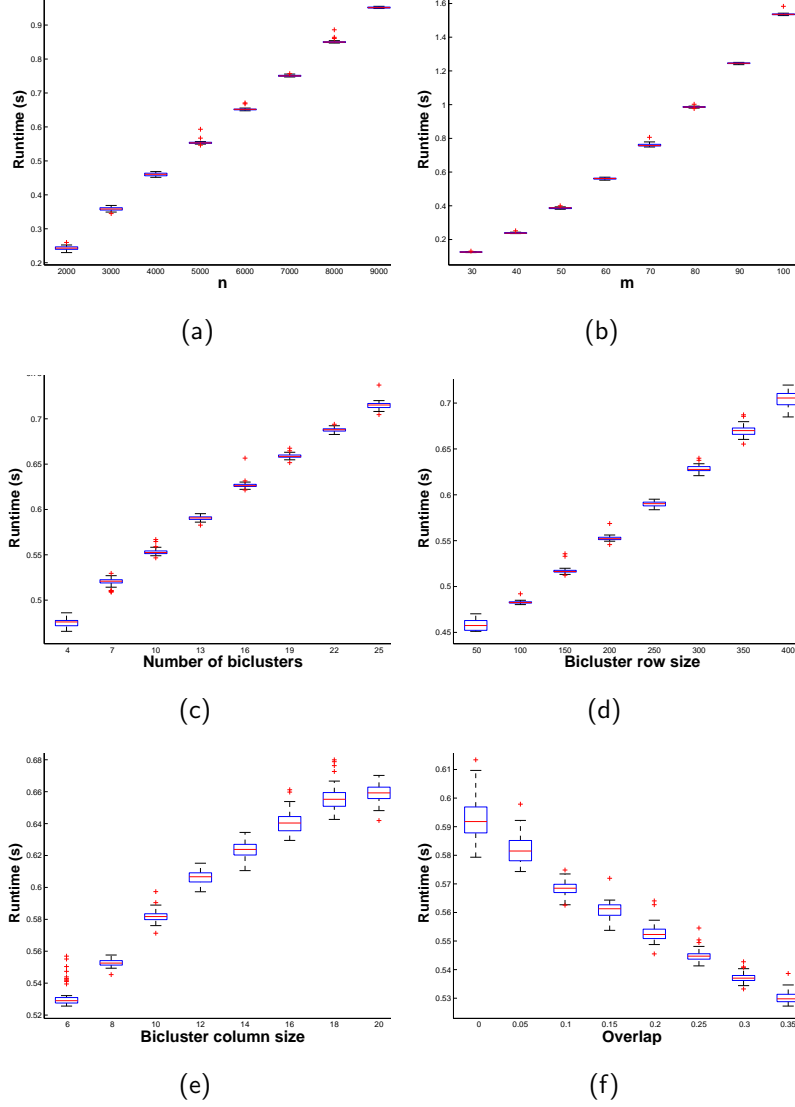


Figure 6: Results of the performance of RIn-Close_CHV_P when varying (a) the number n of rows of the dataset, (b) the number m of columns of the dataset, (c) the number of biclusters in the dataset, (d) the bicluster row size, (e) the bicluster column size, and (f) the overlap.

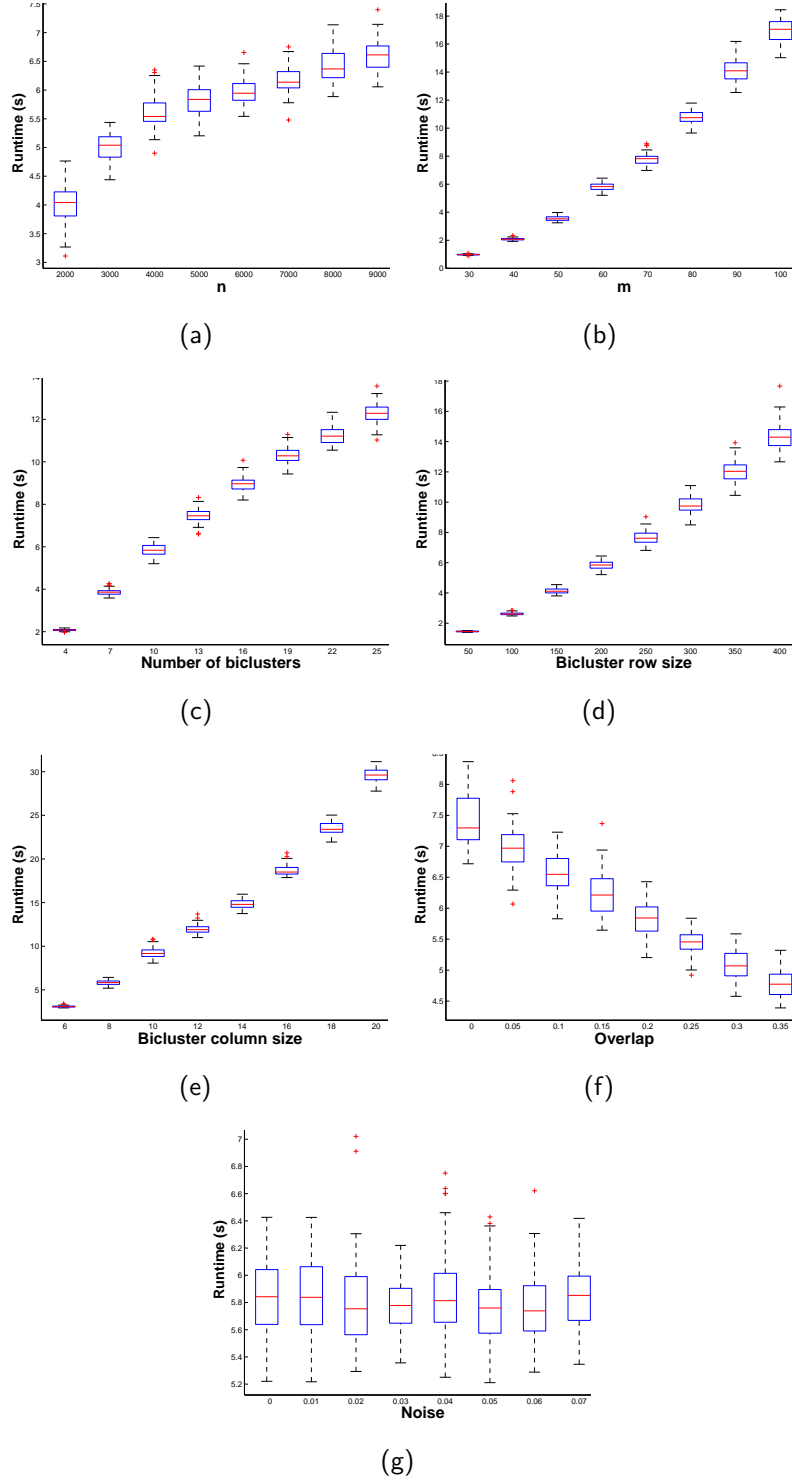


Figure 7: Results of the performance of RIn-Close_CHV when varying (a) the number n of rows of the dataset, (b) the number m of columns of the dataset, (c) the number of biclusters in the dataset, (d) the bicluster row size, (e) the bicluster column size, (f) the overlap, and (g) the noise.

Table 5: Datasets description.

Name	Dimension	Organism
Yeast	2882×17	<i>Saccharomyces cerevisiae</i>
GDS232	589×23	<i>Homo sapiens</i>
GDS750	3456×13	<i>Saccharomyces cerevisiae</i>
GDS4085	1133×19	<i>Homo sapiens</i>

6.2. Sensitivity Analysis

This experiment aims to test RIn-Close’s sensitivity to the parameters ϵ and *minRow*. We ran RIn-Close for four microarray gene expression datasets: Yeast² [17], GDS232³ [55], GDS750³ [51] and GDS4085³ [39]. The former dataset, Yeast, were preprocessed by Cheng and Church [16], we just threw away the two genes with missing values. The last three datasets were preprocessed by us. For each one of them, we remove the empty spots; we threw away the data for any genes where one or more expression levels were not measured; we filtered out genes with small variance over time; and we scale the data of each column to integers between 0 and 1000. Table 5 shows more information about these datasets. We ran RIn-Close 50 times to compute the average runtime, and we looked for biclusters with at least 3 columns.

Figs. 8 and 9 shows respectively the sensitivity of RIn-Close_CVC and RIn-Close_CHV to the parameter ϵ . Usually, the number of biclusters, the runtime, the coverage, and the global overlap increases with ϵ . The only exception is the global overlap of the dataset Yeast in Fig. 8(d). The coverage will always increase with ϵ , because all portions of the dataset explored with $\epsilon = x$, will be explored with $\epsilon > x$, as stated in Property 2.3 (see Subsection 2.4). However, as we stated in Subsection 2.4, the number of biclusters will not always increase with ϵ . The global overlap depends on the coverage and the number of biclusters. If we increase ϵ and find more biclusters, but we explore pretty much the same portions of the dataset, the global overlap will increase. On the other hand, if these new biclusters bring a significant gain in coverage, the global overlap tends to decrease. Therefore, when the global overlap’s growth rate is higher than the coverage’s growth rate, it indicates that we are finding new biclusters in portions of the dataset explored with lower values of ϵ . Oliveira *et al.* [63] illustrated how the noise is responsible for fragmenting each true bicluster into many with high overlapping. As it complicates the analysis of the results, the aggregation of these biclusters is recommended [63, 78]. Given that the runtime is proportional to the number of biclusters, the choice of ϵ must consider the gain in the coverage and the gain in the global overlap. If we only have a considerable gain in the global overlap, then it might be more practicable to use a lower ϵ .

²<http://arep.med.harvard.edu/biclustering>, last accessed 04/09/2015

³<http://www.ncbi.nlm.nih.gov>, last accessed 04/09/2015

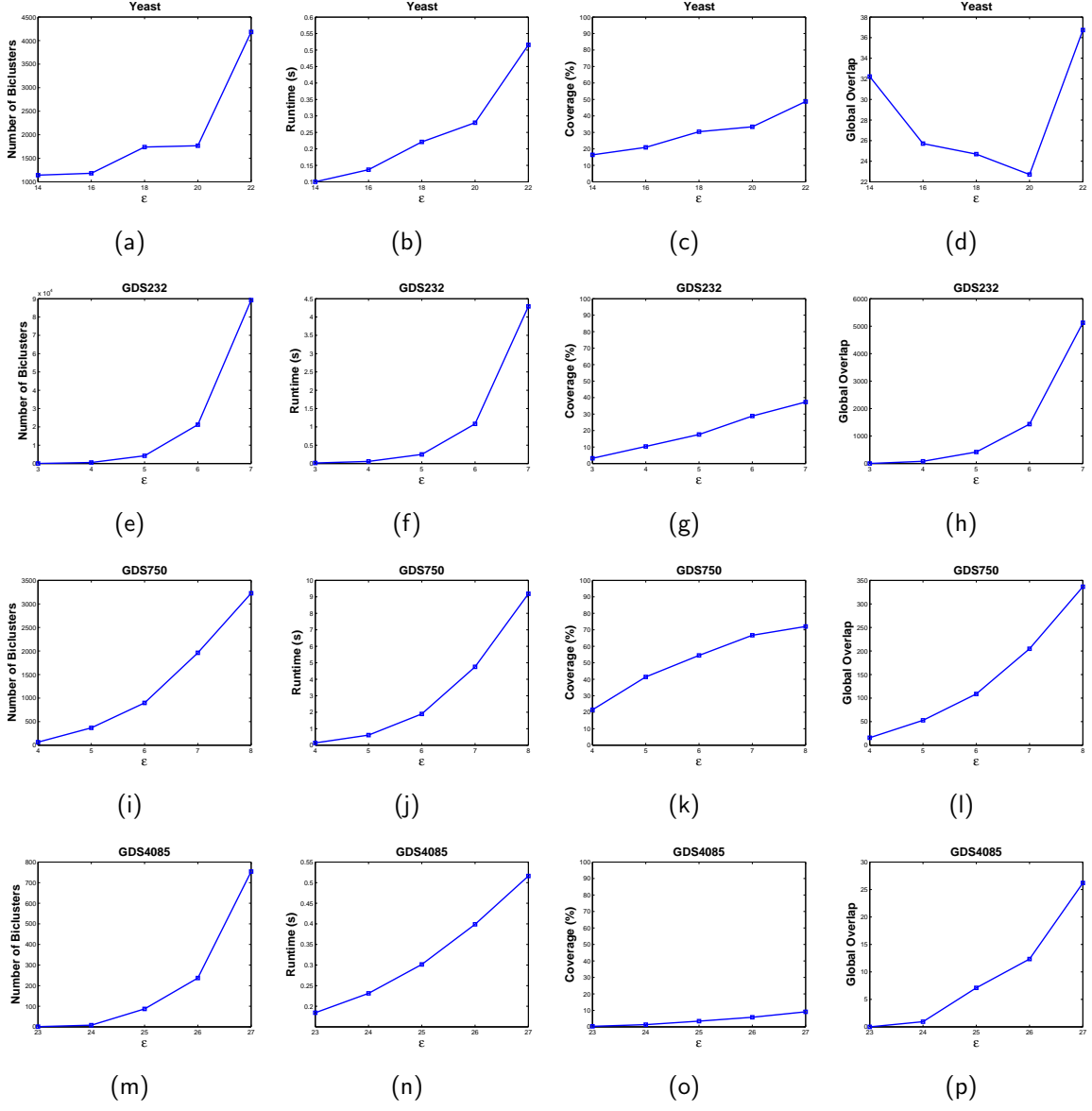


Figure 8: Results of RIn-Close.CVC's sensitivity to the parameter ϵ . The parameter *minRow* was set to: 144 for Yeast; 59 for GDS232; 795 for GDS750; and 23 for GDS4085.

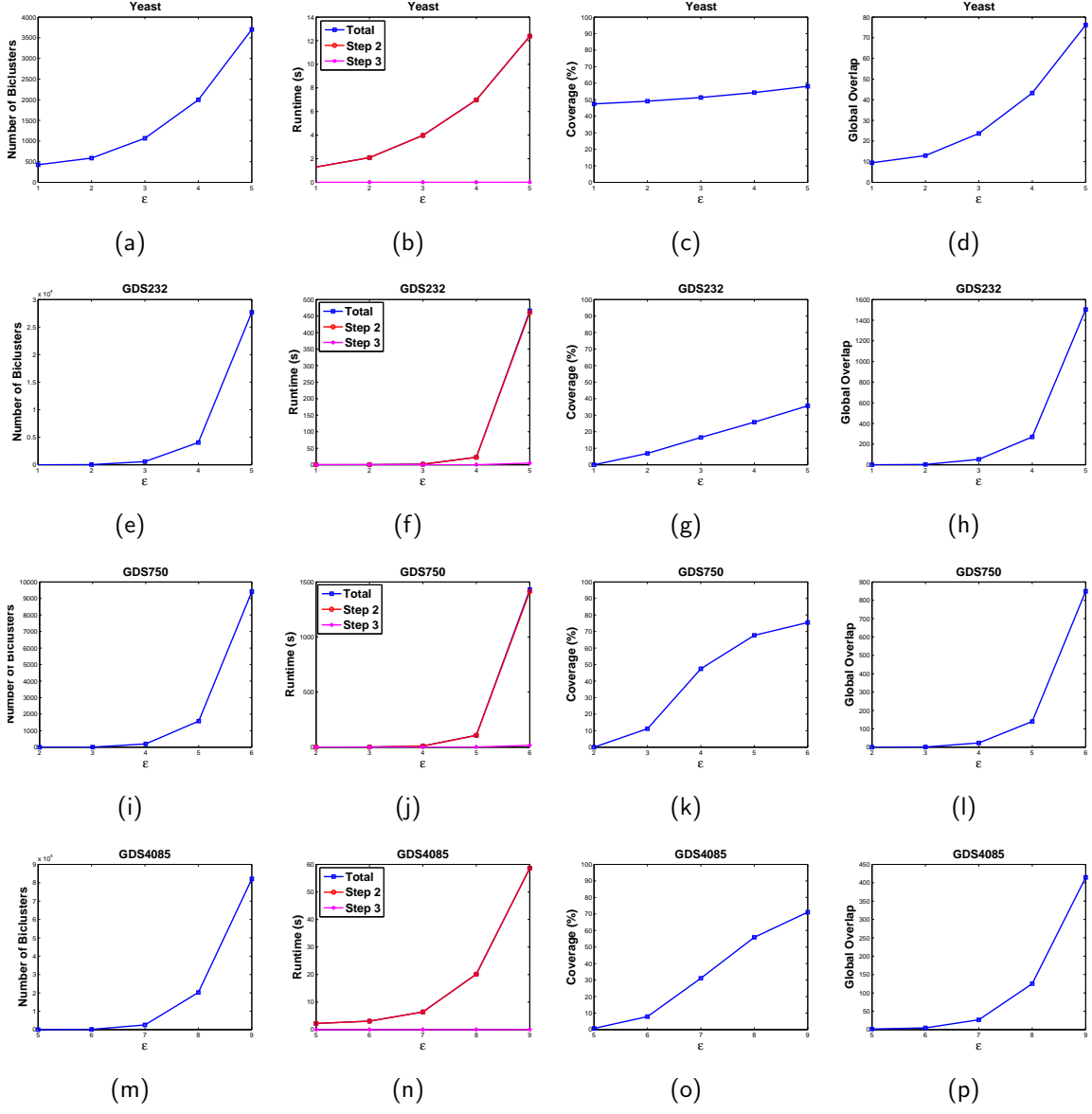


Figure 9: Results of RIn-Close.CHV's sensitivity to the parameter ϵ . The parameter *minRow* was set to: 144 for Yeast; 59 for GDS232; 795 for GDS750; and 23 for GDS4085.

Figs. 10 and 11 shows respectively the sensitivity of RIn-Close_CVC and RIn-Close_CHV to the parameter *minRow*. The parameter *minRow* also has a strong influence in the computational cost of RIn-Close. The higher its value, the smaller the search space for enumerating biclusters. Again, we see that the choice of *minRow* must consider the relation between coverage and global overlap. If a larger value of *minRow* has a small impact on the coverage and a significant impact on the global overlap, certainly it is a reasonable choice.

As we observed with this experiment, RIn-Close parameters can be set in a way that mitigates the explosion in the number of biclusters, with similar impact on the computational cost.

6.3. Comparison with Baselines

With this experiment, we are going to demonstrate that well-known heuristic-based approaches can fail spectacularly when trying to identify the existing biclusters in a dataset. We claim that the results to be presented turn to be a strong motivation for adopting enumerative algorithms, such as the ones proposed in this paper. The dataset is also carefully designed so that we can propose a suitable set of parameters for the heuristic-based approaches, given that we are aware of the main attributes of the existing biclusters. So, the disastrous behavior of the heuristic-based approaches can not be attributed to an unfortunate parameterization. We tested three heuristics that are specialized to mine CHV biclusters: CC, FLOC, and ROCC. These contenders were briefly described in Section 4. We chose to perform this experiment with the CHV type of biclusters because this is the most general type addressed in this work.

For this experiment, we used the 50 synthetic datasets described in Section 6.1 with the default parameters (i.e., $n = 5000$, $m = 60$, number of biclusters = 10, bicluster row size = 200, bicluster column size = 8, overlap = 0.2, and Gaussian noise with $\mu = 0$ and $\sigma = 0.01$). These datasets represent a particular and controlled scenario, i.e., there is a very clear boundary between what should and what should not be part of a bicluster. Possibly, the boundaries are not so accurate in real-world applications. But in this way, these dataset allows us to clearly determine the parameters of the biclustering algorithms, and find out what they are able to mine when looking for the original biclusters. For CC and FLOC, we set the value of δ for each dataset considering its largest bicluster MSR. For both, the number of biclusters to be mined were set to 10. CC's threshold for multiple node deletion α was set to 1.2 (the value suggested by the authors). For FLOC, we set the probability to add a row/column to a seed (initial) bicluster based on the proportion of the minimum number of rows/columns of a bicluster and the total number of rows/columns in the dataset. So, we set these parameters to 0.04 and 0.13, respectively. For ROCC, we set $sr = 1586$ and $sc = 17$ because these are the number of distinct rows/columns covered by the biclusters. Based on the fraction of the number of rows/columns of a dataset over the number of rows/columns of a bicluster, we set $k = 25$ and $l = 8$.

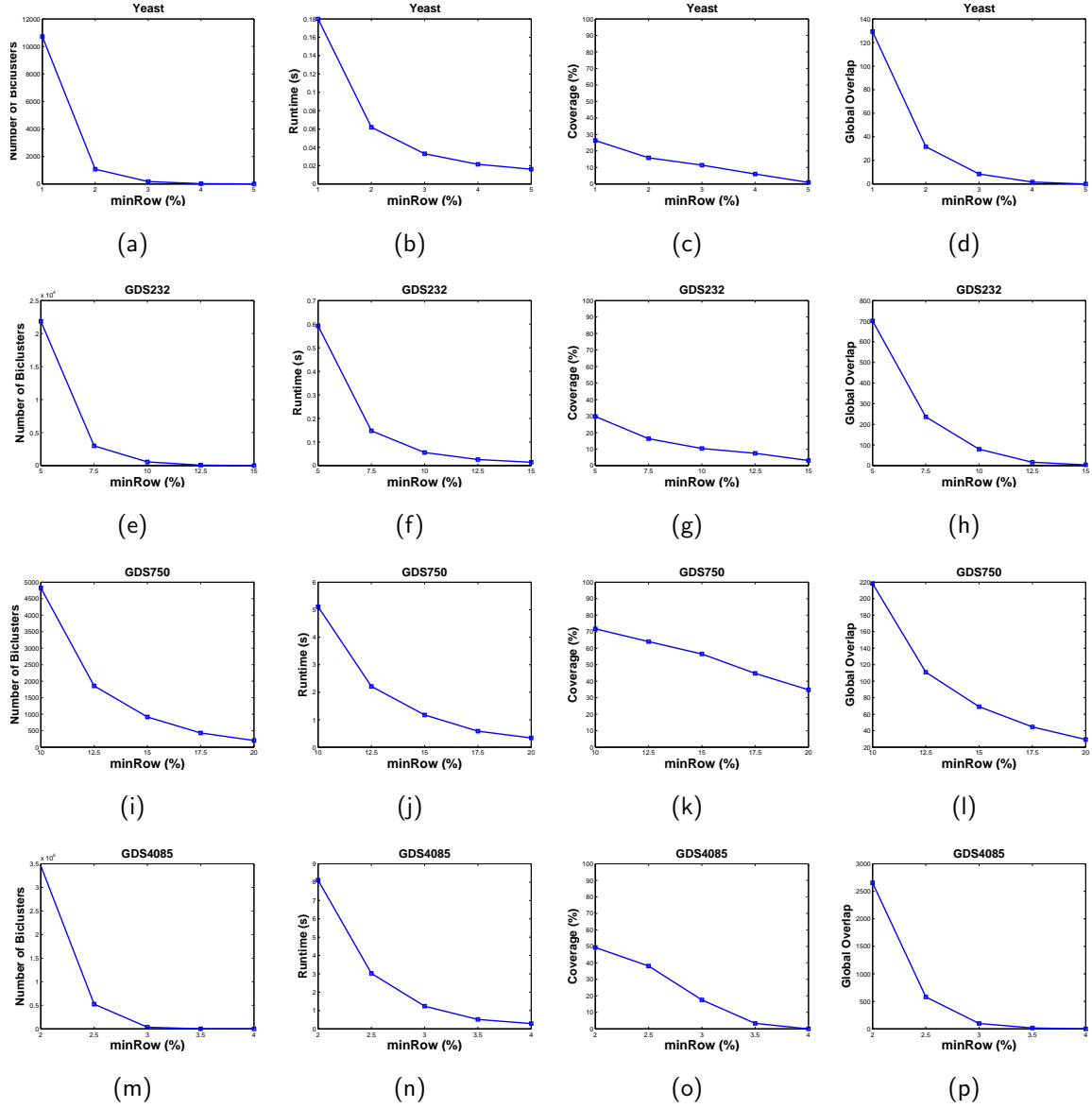


Figure 10: Results of RIn-Close.CVC's sensitivity to the parameter $minRow$. The parameter ϵ was set to: 5 for Yeast; 4 for GDS232; 4 for GDS750; and 37 for GDS4085.

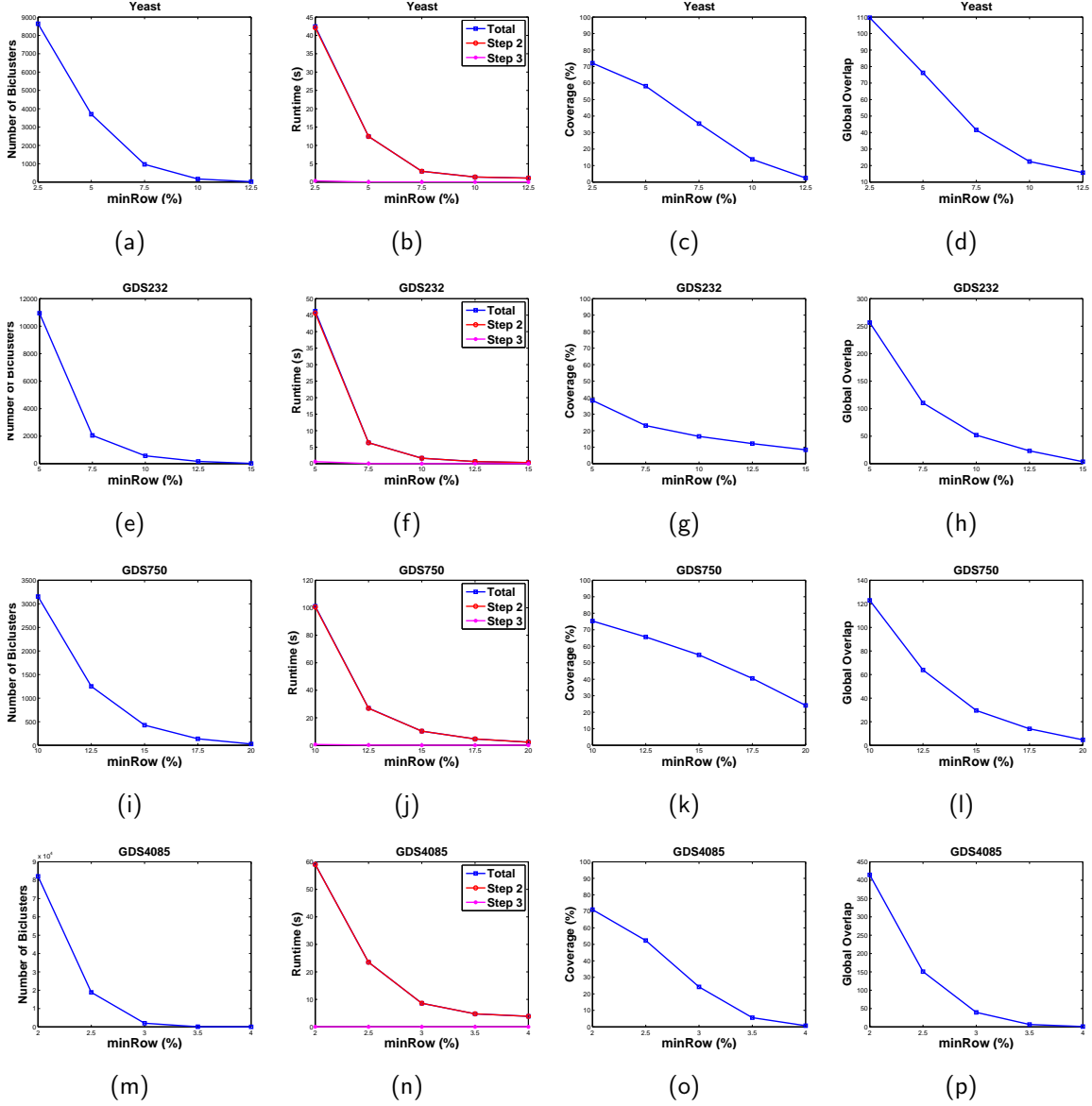


Figure 11: Results of RIn-Close.CHV's sensitivity to the parameter $minRow$. The parameter ϵ was set to: 5 for Yeast; 3 for GDS232; 3 for GDS750; and 9 for GDS4085.

Table 6: Results of the comparison with baselines.

	Precision	Recall
RIn-Close	1	1
CC	-	0
FLOC	0.0768 (0.0241)	0.0874 (0.0252)
ROCC	0.1831 (0.0355)	0.3845 (0.0655)

Table 6 shows the results of this experiment. CC completely failed in finding the original biclusters. As the values of the parameter δ were very low, CC was unable to find so accurate submatrices. FLOC had a very poor result with low Precision and Recall. As the initial biclusters are generated at random, it is very unlikely that FLOC can improve them to the original ones, that have a very clear boundary between what should and what should not be part of a bicluster, thus having a very low MSR. ROCC had better results than CC and FLOC. ROCC starts with a checkboard biclustering structure containing all rows and columns, so it is expected that ROCC would achieve a better Recall than the others. But its routine to refine this initial solution was not accurate, which led to a low Precision, even though its Precision was better than the others.

As we have seen, although we knew how to choose good parameters for the heuristic-based algorithms, this case study was very challenging to them. On the other hand, RIn-Close easily accomplishes this task.

7. Conclusion

Biclustering is a very powerful data mining technique that overcomes several drawbacks of the well-known clustering technique. Due to its complexity, most of the proposed biclustering algorithms are heuristic-based. Nonetheless, there are several algorithms able to perform (i) efficient, (ii) complete, (iii) correct, and (iv) non-redundant enumeration of all maximal CTV biclusters of ones from a binary data matrix. These enumerative algorithms proved to be very useful and have been applied in various application domains. Nonetheless, the raw data matrix admits integer and/or real values in several other application domains, and to transform it into binary data leads to loss of information. Hence, there are some proposals capable of dealing directly with numerical data matrices, but none of these algorithms to enumerate CVC, CVR, or CHV biclusters keeps these four properties.

In this paper, we proposed a family of algorithms, called RIn-Close, capable of preserving these four properties when enumerating perfect CVC (or CVR) biclusters, perturbed CVC (or CVR) biclusters, and perfect CHV biclusters, and capable of preserving the last three of these properties when enumerating perturbed CHV biclusters. As far as we know, our algorithms are the most complete from the literature.

Our experimental results provided a valuable insight into the scalability of RIn-Close and its sensitivity to the user-defined measure of similarity and minimum number of rows allowed in a bicluster. As the larger the dataset, the greater tends to be the number of biclusters, these parameters are critical to feasibility of the biclustering solution. We also showed that well-known heuristic-based algorithms can fail spectacularly when trying to identify the existing biclusters in a simple and controlled scenario, thus pointing to the necessity of having efficient enumerative biclustering algorithms.

In future works, we are planning to extend the proposal to handle data matrices with missing values, and to enumerate biclusters with coherent evolutions. We also intend to investigate the extension of RIn-Close to enumerate only the top k biclusters in terms of volume, thus reducing RIn-Close computational cost. Still looking at the reduction of their computational cost, we will also implement parallelized versions of RIn-Close algorithms.

Acknowledgments

R. Veroneze and F. J. Von Zuben would like to thank CAPES and CNPq for the financial support. A. Banerjee acknowledges support of NSF grants IIS-1447566, IIS-1422557, CCF-1451986, CNS-1314560, IIS-0953274, IIS-1029711, IIS-0916750, and NASA grant NNX12AQ39A.

References

References

- [1] J. ABELLO, A. J. POGEL, AND L. MILLER, *Breadth first search graph partitions and concept lattices.*, Journal of Universal Computer Science, 10 (2004), pp. 934–954.
- [2] R. AGRAWAL, J. GEHRKE, D. GUNOPULOS, AND P. RAGHAVAN, *Automatic subspace clustering of high dimensional data for data mining applications*, in Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, vol. 27, 1998, pp. 94–105.
- [3] R. AGRAWAL, T. IMIELIŃSKI, AND A. SWAMI, *Mining association rules between sets of items in large databases.*, in Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 1993, pp. 207–216.
- [4] R. AGRAWAL, R. SRIKANT, ET AL., *Fast algorithms for mining association rules*, in Proceedings of the 20th International Conference on Very Large Data Bases, vol. 1215, 1994, pp. 487–499.
- [5] F. ALQADAH, C. K. REDDY, J. HU, AND H. F. ALQADAH, *Biclustering neighborhood-based collaborative filtering method for top-n recommender systems*, Knowledge and Information Systems, (2014), pp. 1–17.

- [6] S. ANDREWS, *In-Close, a fast algorithm for computing formal concepts*, in International Conference on Conceptual Structures, 2009.
- [7] ———, *In-Close2, a high performance formal concept miner.*, in International Conference on Conceptual Structures, 2011, pp. 50–62.
- [8] G. ATLURI, J. BELLAY, G. PANDEY, C. MYERS, AND V. KUMAR, *Discovering coherent value bicliques in genetic interaction data*, in Proceedings of 9th International Workshop on Data Mining in Bioinformatics (BIOKDD10), Citeseer, 2000.
- [9] J. BAIXERIES, M. KAYTOUE, AND A. NAPOLI, *Characterizing functional dependencies in formal concept analysis with pattern structures*, Annals of mathematics and artificial intelligence, 72 (2014), pp. 129–149.
- [10] A. BANERJEE, I. DHILLON, J. GHOSH, S. MERUGU, AND D. S. MODHA, *A generalized maximum entropy approach to bregman co-clustering and matrix approximation*, Journal of Machine Learning Research, 8 (2007), pp. 1919–1986.
- [11] A. BEN-DOR, B. CHOR, R. KARP, AND Z. YAKHINI, *Discovering local structure in gene expression data: the order-preserving submatrix problem*, Journal of computational biology, 10 (2003), pp. 373–384.
- [12] J. BESSON, C. ROBARDET, L. DE RAEDT, AND J.-F. BOULICAUT, *Mining bi-sets in numerical data*, in Knowledge Discovery in Inductive Databases, Springer, 2007, pp. 11–23.
- [13] C. BRON AND J. KERBOSCH, *Algorithm 457: finding all cliques of an undirected graph*, Communications of the ACM, 16 (1973), pp. 575–577.
- [14] S. BUSYGIN, O. PROKOPYEV, AND P. M. PARDALOS, *Biclustering in data mining*, Computers & Operations Research, 35 (2008), pp. 2964–2987.
- [15] F. CAZALS AND C. KARANDE, *A note on the problem of reporting maximal cliques*, Theoretical Computer Science, 407 (2008), pp. 564–568.
- [16] Y. CHENG AND G. CHURCH, *Biclustering of expression data*, in Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology, vol. 1, 2000, pp. 93–103.
- [17] R. J. CHO, M. J. CAMPBELL, E. A. WINZELER, L. STEINMETZ, A. CONWAY, L. WODICKA, T. G. WOLFSBERG, A. E. GABRIELIAN, D. LANDSMAN, D. J. LOCKHART, ET AL., *A genome-wide transcriptional analysis of the mitotic cell cycle*, Molecular Cell, 2 (1998), pp. 65–74.

- [18] V. CODOCEDO AND A. NAPOLI, *Bicluster enumeration using formal concept analysis*, in What formal concept analysis can do for artificial intelligence?(FCA4AI 2014) Workshop at ECAI 2014, 2014.
- [19] ———, *Lattice-based biclustering using partition pattern structures*, in ECAI 2014: 21st European Conference on Artificial Intelligence, vol. 263, IOS Press, 2014, p. 213.
- [20] A. COLANTONIO, R. DI PIETRO, A. OCELLO, AND N. VERDE, *Abba: Adaptive bicluster-based approach to impute missing values in binary matrices*, in Proceedings of the 2010 ACM Symposium on Applied Computing, 2010, pp. 1026–1033.
- [21] P. DE CASTRO, F. DE FRANÇA, H. FERREIRA, AND F. VON ZUBEN, *Applying biclustering to perform collaborative filtering*, in 7th International Conference on Intelligent Systems Design and Applications, 2007, pp. 421–426.
- [22] F. O. DE FRANÇA, G. P. COELHO, AND F. J. VON ZUBEN, *Predicting missing values with biclustering: A coherence-based approach*, Pattern Recognition, 46 (2013), pp. 1255–1266.
- [23] M. DEODHAR, G. GUPTA, J. GHOSH, H. CHO, AND I. DHILLON, *A scalable framework for discovering coherent co-clusters in noisy data*, in Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 241–248.
- [24] I. S. DHILLON, *Co-clustering documents and words using bipartite spectral graph partitioning*, in Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 269–274.
- [25] D. EPPSTEIN, M. LÖFFLER, AND D. STRASH, *Listing all maximal cliques in sparse graphs in near-optimal time*, in Algorithms and Computation, Springer, 2010, pp. 403–414.
- [26] B. GANTER, *Two basic algorithms in concept analysis*, tech. report, FB4-Preprint No. 831, 1984.
- [27] B. GANTER AND S. O. KUZNETSOV, *Pattern structures and their projections*, in Conceptual Structures: Broadening the Base, Springer, 2001, pp. 129–142.
- [28] B. GANTER AND R. WILLE, *Formal concept analysis: mathematical foundations*, Springer-Verlag New York, Inc., 1997.
- [29] B. GAUME, E. NAVARRO, AND H. PRADE, *A parallel between extended formal concept analysis and bipartite graphs analysis*, in Computational Intelligence for Knowledge-Based Systems Design, Springer, 2010, pp. 270–280.

- [30] A. GÉLY, L. NOURINE, AND B. SADI, *Enumeration aspects of maximal cliques and bicliques*, Discrete Applied Mathematics, 157 (2009), pp. 1447–1459.
- [31] B. GOETHALS, *Survey on frequent pattern mining*, Univ. of Helsinki, (2003).
- [32] K. GOUDA AND M. J. ZAKI, *Genmax: An efficient algorithm for mining maximal frequent itemsets*, Data Mining and Knowledge Discovery: An International Journal, 11 (2005), pp. 223–242.
- [33] J. GUAN, Y. GAN, AND H. WANG, *Discovering pattern-based subspace clusters by pattern tree*, Knowledge-Based Systems, 22 (2009), pp. 569–579.
- [34] J. HARTIGAN, *Direct clustering of a data matrix*, Journal of the American Statistical Association, (1972), pp. 123–129.
- [35] R. HENRIQUES AND S. C. MADEIRA, *Bicpam: Pattern-based biclustering for biomedical data analysis*, Algorithms for Molecular Biology, 9 (2014), p. 27.
- [36] D. HORTA AND R. J. CAMPELLO, *Similarity measures for comparing biclusterings*, Computational Biology and Bioinformatics, IEEE/ACM Transactions on, 11 (2014), pp. 942–954.
- [37] J. IHMELS, G. FRIEDLANDER, S. BERGMANN, O. SARIG, Y. ZIV, AND N. BARKAI, *Revealing modular organization in the yeast transcriptional network*, Nature genetics, 31 (2002), pp. 370–377.
- [38] N. K. V. J. K. GUPTA, S. SINGH, *Mtba: Matlab toolbox for biclustering analysis*, IEEE, 2013, pp. 94–97.
- [39] S. JULIEN, A. IVETIC, A. GRIGORIADIS, D. QIZE, B. BURFORD, D. SPROVIERO, G. PICCO, C. GILLET, S. L. PAPP, L. SCHAFFER, ET AL., *Selectin ligand sialyl-lewis x antigen drives metastasis of hormone-dependent breast cancers*, Cancer research, 71 (2011), pp. 7683–7693.
- [40] M. KAYTOUE, Z. ASSAGHIR, A. NAPOLI, AND S. O. KUZNETSOV, *Embedding tolerance relations in formal concept analysis: an application in information fusion*, in Proceedings of the 19th ACM international conference on Information and knowledge management, ACM, 2010, pp. 1689–1692.
- [41] M. KAYTOUE, V. CODOCEDO, J. BAIXERIES, AND A. NAPOLI, *Three related fca methods for mining biclusters of similar values on columns*, in Proceedings of the Eleventh International Conference on Concept Lattices and Their Applications, Kosice, Slovakia, October 7-10, 2014, 2014.
- [42] M. KAYTOUE, S. O. KUZNETSOV, J. MACKO, AND A. NAPOLI, *Biclustering meets triadic concept analysis*, Annals of Mathematics and Artificial Intelligence, (2013), pp. 1–25.

- [43] M. KAYTOUE, S. O. KUZNETSOV, AND A. NAPOLI, *Biclustering numerical data in formal concept analysis*, in 9th International Conference on Formal Concept Analysis, 2011, pp. 135–150.
- [44] I. KOCH, *Enumerating all connected maximal common subgraphs in two graphs*, Theoretical Computer Science, 250 (2001), pp. 1–30.
- [45] P. KRAJCA, J. OTRATA, AND V. VYCHODIL, *Advances in algorithms based on cbo*, in Proceedings of the 8th International Conference on Concept Lattices and Their Applications, vol. 672, 2010, pp. 325–337.
- [46] H.-P. KRIEGEL, P. KRÖGER, AND A. ZIMEK, *Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering*, ACM Transactions on Knowledge Discovery from Data (TKDD), 3 (2009), p. 1.
- [47] S. KUZNETSOV, *Mathematical aspects of concept analysis*, Journal of Mathematical Sciences, 80 (1996), pp. 1654–1698.
- [48] S. KUZNETSOV, *Learning of simple conceptual graphs from positive and negative examples*, in Principles of Data Mining and Knowledge Discovery, 1999, pp. 384–391.
- [49] L. LAKHAL AND G. STUMME, *Efficient mining of association rules based on formal concept analysis*, in Formal concept analysis: Foundations and Applications, Springer, 2005, pp. 180–195.
- [50] L. LAZZERONI AND A. OWEN, *Plaid models for gene expression data*, Statistica Sinica, 12 (2002), pp. 61–86.
- [51] J. H. LEBER, S. BERNALES, AND P. WALTER, *Ire1-independent gain control of the unfolded protein response*, PLoS biology, 2 (2004), p. e235.
- [52] F. LEHMANN AND R. WILLE, *A triadic approach to formal concept analysis*, Springer, 1995.
- [53] B. LIU, W. HSU, AND Y. MA, *Integrating classification and association rule mining*, in Proceedings of the fourth international conference on knowledge discovery and data mining, 1998.
- [54] Y. LIU, Q. GU, J. P. HOU, J. HAN, AND J. MA, *A network-assisted co-clustering algorithm to discover cancer subtypes based on gene expression*, BMC bioinformatics, 15 (2014), p. 37.
- [55] T. J. MACDONALD, K. M. BROWN, B. LAFLEUR, K. PETERSON, C. LAWLOR, Y. CHEN, R. J. PACKER, P. COGEN, AND D. A. STEPHAN, *Expression profiling of medulloblastoma: Pdgfra and the ras/mapk pathway as therapeutic targets for metastatic disease*, Nature genetics, 29 (2001), pp. 143–152.

- [56] S. MADEIRA AND A. OLIVEIRA, *Biclustering algorithms for biological data analysis: a survey*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, 1 (2004), pp. 24–45.
- [57] S. C. MADEIRA AND A. L. OLIVEIRA, *A polynomial time biclustering algorithm for finding approximate expression patterns in gene expression time series*, Algorithms for Molecular Biology, 4 (2009), p. 8.
- [58] K. MAKINO AND T. UNO, *New algorithms for enumerating all maximal cliques*, in Proceedings of the 9th Scandinavian Workshop on Algorithm Theory, 2004, pp. 260–272.
- [59] R. MARTINEZ, C. PASQUIER, AND N. PASQUIER, *Genminer: mining informative association rules from genomic data*, in Bioinformatics and Biomedicine, 2007. BIBM 2007. IEEE International Conference on, IEEE, 2007, pp. 15–22.
- [60] B. MIRKIN, *Mathematical Classification and Clustering*, Springer, 1996.
- [61] Y. OKADA, W. FUJIBUCHI, AND P. HORTON, *A biclustering method for gene expression module discovery using a closed itemset enumeration algorithm*, IPSJ Digital Courier, 3 (2007), pp. 183–192.
- [62] Y. OKADA, K. OKUBO, P. HORTON, AND W. FUJIBUCHI, *Exhaustive search method of gene expression modules and its application to human tissue data*, IAENG international journal of computer science, 34 (2007).
- [63] S. OLIVEIRA, R. VERONEZE, AND F. J. VON ZUBEN, *On bicluster aggregation and its benefits for enumerative solutions*, in 11th International Conference on Machine Learning and Data Mining, 2015, pp. 135–150.
- [64] G. PANDEY, G. ATLURI, M. STEINBACH, C. L. MYERS, AND V. KUMAR, *An association analysis approach to biclustering*, in Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 677–686.
- [65] T. PANG-NING, M. STEINBACH, AND V. KUMAR, *Introduction to Data Mining*, Pearson, 2005.
- [66] J. PEI, X. ZHANG, M. CHO, H. WANG, AND P. S. YU, *Maple: A fast algorithm for maximal pattern-based clustering*, in 3th IEEE International Conference on Data Mining, 2003, pp. 259–266.
- [67] A. SERIN AND M. VINGRON, *Debi: Discovering differentially expressed biclusters using a frequent itemset approach.*, Algorithms for Molecular Biology, 6 (2011), p. 18.
- [68] P. SYMEONIDIS, A. NANOPOULOS, A. PAPADOPOULOS, AND Y. MANOLOPOULOS, *Nearest-biclusters collaborative filtering with constant values*, Advances in Web Mining and Web Usage Analysis, (2007), pp. 36–55.

- [69] P. SYMEONIDIS, A. NANOPOULOS, A. N. PAPADOPOULOS, AND Y. MANOLOPOULOS, *Nearest-biclusters collaborative filtering based on constant and coherent values*, Information retrieval, 11 (2008), pp. 51–75.
- [70] T. UNO, M. KIYOMI, AND H. ARIMURA, *Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets*, in Workshop on Frequent Itemset Mining Implementations, vol. 19, 2004, p. 30.
- [71] R. VERONEZE, F. DE FRANCA, AND F. VON ZUBEN, *Assessing the performance of a swarm-based biclustering technique for data imputation*, in IEEE Congress on Evolutionary Computation, 2011, pp. 386–393.
- [72] H. WANG, F. CHU, W. FAN, P. S. YU, AND J. PEI, *A fast algorithm for subspace clustering by pattern similarity*, in Proceedings of the 16th International Conference on Scientific and Statistical Database Management, 2004, pp. 51–60.
- [73] H. WANG, W. WANG, J. YANG, AND P. S. YU, *Clustering by pattern similarity in large data sets*, in Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, 2002, pp. 394–405.
- [74] J. YANG, H. WANG, W. WANG, AND P. YU, *Enhanced biclustering on expression data*, in Proceedings of the 3th IEEE Symposium on Bioinformatics and Bioengineering, 2003, pp. 321–327.
- [75] M. J. ZAKI, *Generating non-redundant association rules*, in Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2000, pp. 34–43.
- [76] M. J. ZAKI AND C.-J. HSIAO, *Charm: An efficient algorithm for closed itemset mining*, in Proceedings of the 2002 SIAM International Conference on Data Mining, vol. 2, 2002, pp. 457–473.
- [77] M. J. ZAKI, S. PARTHASARATHY, M. OGIHARA, AND W. LI, *New algorithms for fast discovery of association rules*, in 3rd International Conference on Knowledge Discovery and Data Mining, 1997, pp. 283–286.
- [78] L. ZHAO AND M. J. ZAKI, *Microcluster: Efficient deterministic biclustering of microarray data*, Intelligent Systems, 20 (2005), pp. 40–49.

Rosana Veroneze is a PhD candidate in Electrical and Computer Engineering at the University of Campinas (Unicamp). Her research interests includes data mining and machine learning areas.

Arindam Banerjee is an Associate Professor at the Department of Computer and Engineering and a Resident Fellow at the Institute on the Environment at the University of Minnesota, Twin Cities. His research interests are in machine learning, data mining, convex analysis and optimization, and their applications in complex real-world problems.

Fernando J. Von Zuben is an Associate Professor at the Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, University of Campinas (Unicamp). The main topics of his research are computational intelligence, bioinspired computing, multivariate data analysis, and machine learning.